Universidade Federal do Rio de Janeiro

State Pre-Selection: a clustering approach to speed up SDDP

Ricardo Turano Figueiredo

Rio de Janeiro
Janeiro de 2023

Universidade Federal do Rio de Janeiro

State Pre-Selection: a clustering approach to speed up SDDP

Ricardo Turano Figueiredo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Advisor: Bernardo Freitas Paulo da Costa

Rio de Janeiro
Janeiro de 2023

Turano Figueiredo, Ricardo

State Pre-Selection: a clustering approach to speed up SDDP / Ricardo Turano Figueiredo. - Rio de Janeiro: UFRJ/IM, 2023.

xii, 92f.: il.; 29,7cm.

Orientador: Bernardo Freitas Paulo da Costa

Dissertação (Mestrado) – UFRJ/IM/Programa de Pós-Graduação em Matemática, 2023.

Referências Bibliográficas: f. 91–92

a. Multi-Stage Stochastic Optimization. b. Cutting-plane method. c. SDDP. d. Cut Selection. e. Long-term energy planning. I. Freitas Paulo da Costa, Bernardo. II. Universidade Federal do Rio de Janeiro, Programa de Pós-Graduação em Matemática. III. Título.

State Pre-Selection: a clustering approach to speed up SDDP

Ricardo Turano Figueiredo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática, Instituto de Matemática da Universidade Federal do Rio de Janeiro (UFRJ), como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Aprovada por:

Bernardo Freitas Paulo da Costa

Prof. Bernardo Freitas Paulo da Costa
(Orientador)

Prof. Andy Philpott

Documento assinado digitalmente
Erlon Cristian Finardi
Data: 26/01/2023 15:34:33-0300
CPF: ***.364.749-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Erlon Cristian Finardi

Documento assinado digitalmente
FABIO ANTONIO TAVARES RAMOS
Data: 27/01/2023 07:25:55-0300
Verifique em https://verificador.iti.br

Prof. Fabio Antonio Tavares Ramos

Rio de Janeiro
Janeiro, 2023

Acknowledgments

Although the content of this work is written entirely in English, the following words are written in Portuguese, my native language. This is done with the hope that the acknowledged people may know how important they were for the process of writing this thesis.

Gostaria de começar agradecendo a minha esposa Shirley de Fátima da Silva Gomes. Comigo desde 2017, sempre esteve ao meu lado me apoiando quando preciva de um empurrão e me freando quando eu precisava. Minha parceira, amiga, companheira em meus momentos mais dificeis. Uma mulher que sempre me inspira a ser melhor, me ensinou e ainda ensina todos os dias como ser um ser humano mais completo. Se eu pude escrever cada página desse trabalho eu devo a você, tanto por não me deixar desistir, quanto por compreender quando eu precisei postergar nossos planos para ter mais tempo para esse trabalho. Meu amor, muito obrigado, eu espero que você saiba que você todo dia é sempre meu primeiro motivo para sorrir.

Em seguida, eu gostaria de agradecer muito ao meu orientador, Bernardo Freitas Paulo da Costa. Mais que um professor, mais que um orientador, o senhor foi e é um grande amigo. A compreensão, carinho e compaixão que o senhor teve comigo nesses anos desde antes da graduação nunca poderão ser retribuídas na mesma quantidade. O senhor me orientou mais do que só na matemática, me orientou na vida. No momento em que eu mais me vi pensando em desistir da matemática, foi o senhor que me abraçou e viu um potencial em mim que nem mesmo eu via mais. Obrigado por ter aberto as portas para mim, obrigado por ter permitido um mestrado tranquilo dentro do possível e obrigado por ter me apresentado o grupo de filosofia ao qual agradecerei em breve.

Agora, gostaria de agradecer a meus pais (José Ricardo e Ceumar), minha madrasta (Carmen), meus avós (Heloá, Luiz, Cilio, Alice e Araujo), meus irmãos (João, Pedro, Camila e Victória), minha prima (Carolina), meus tios (João e Luiza Maria). Primeiro, por ter me aturado todos esses anos. Sei que não sou fácil, sei que sou fechado e meio injusto ou duro com vocês às vezes. Por isso, mais do que um obrigado eu quero pedir desculpas. Todos vocês me ajudaram do jeito que puderam, mesmo que não saibam disso. Nesse bonde dos parentes quero agradecer também aos parentes da minha esposa, em especial à minha sogrinha (Maria Teresa) que é como uma mãe para mim.

Além disso, expresso meus profundos agradecimentos aos participantes da minha banca (professor Andy Philpott, professor Erlon Finardi e professor Fábio Ramos). Na mesma medida, gostaria de agradecer a todos os professores e funcionários das instituições de ensino que frequentei. Em especial gostaria de agradecer o trio de professores Bernardo Freitas, Fábio Ramos e Monique Carmona da UFRJ, aos quais perturbei para resolver inúmeros pepinos e sempre me ajudaram. Gostaria também de agradecer especialmente à professora Nara Barat, minha coordenadora de matemática no Colégio de São Bento, que me apresentou à Matemática Aplicada da UFRJ.

Começo agora agradecer a minhas famílias. Esclareço que ao longo dos anos eu tive várias famílias que me acolheram em diversos ambientes e que foram importantes para minha formação como matemático e ser humano.

Família Matemática Aplicada / ABC-116: A vocês meu profundo carinho e respeito. Desde meu primeiro semestre me senti em casa com vocês e nos últimos da graduação isso ficou até mais evidente (entendedores entenderão). Na matemática aplicada encontrei professores que faziam questão de consultar os alunos sobre o andamento do curso. Encontrei também veteranos que não queriam deixar nenhum calouro para trás e calouros que faziam questão de cuidar do curso. Além dos agregados que viviam o curso como se estivessem no mesmo. Toda essa estrutura foi fundamental para que eu passasse pela graduação de forma mais tranquila. Gostaria de agradecer em especial a alguns nomes (com alguns títulos/apelidos da minha cabeça) que ainda não foram citados nesse texto: Bruno Lima Netto (Grifinória), Ivani Ivanova (A Ivani), Iago Leal (O alto), Rodrigo Peregrino (Digão), Gabriella Radke (Gabig), Leonardo Gama (O monitor), Tiago Vital (França), Hugo Carvalho (O matmúsico), Aloizio Macedo (Se ele não soube ninguém sabe), Vitor Luiz (O Vitor inteligente), Gabriel Sanfins (Baby), Rafael Klausner (O meu salvador do SDDP), Ruan Felipe (FFXIV), Gabriel Picanço (Pikanço), Vitor Hugo (Forró), Alexandre Moreira (Caveira), Matheus Fontoura (Jesus), Filipe Cabral (O primeiro dos otimizadores), Jonathas Ferreira (Bangu), Guilherme Monteiro ("Meu querido"), Renan (Gênio).

Família Hora do Play / Filosofia Amiga: A vocês minha admiração e gratidão. Essa família me ensinou muito mais do que eu achei que eu poderia aprender fora da matemática. Vocês todos são pilar central da minha visão de mundo hoje em dia. Me ensinaram muito sobre a vida, desde rever preconceitos a saber apreciar um bom filme. Gostaria que vocês soubessem que são todos parte fundamentais da minha vida e portanto indispensáveis nessa caminhada do mestrado. Muito obrigado, Suely, Joari, Bernardo, Conrado, Nathalie, Vitinho, Maíra, Tavares, Eberhart, Mário, Juliana, Marcos, Gabriel, Shirley e Tatiana. Em especial, gostaria de agradecer ao Joari pela ajuda com o resumo da dissertação.

Por fim, gostaria de agradecer mais algumas famílias que estiveram comigo nessa jornada. O grupo dos colegas da escola (pessoal do Koe Cara), os amigos que fiz no WoW, o pessoal da Point HQ e o grupo de pesquisa do ONS.

State Pre-Selection: a clustering approach to speed up SDDP

Ricardo Turano Figueiredo

Advisor: Bernardo Freitas Paulo da Costa

Abstract

This dissertation proposes a technique to help solve optimization problems using data obtained by solving similar problems. The development of this technique, hereafter called *State Pre-Selection*, will be illustrated considering a convex stochastic multistage optimization problem. The proposed technique will be applied in a case study based on the Brazilian energy planning problem, and the results will be presented and discussed.

Keywords: Multi-Stage Stochastic Optimization, Cutting-plane method, SDDP, Cut Selection, Long-term energy planning.

Rio de Janeiro
Janeiro de 2023

State Pre-Selection: a clustering approach to speed up SDDP

Ricardo Turano Figueiredo

Orientador: Bernardo Freitas Paulo da Costa

Resumo

Essa dissertação propõe uma técnica para ajuda a resolve problemas de otimização utilizando dados obtidos a partir da resolução de problemas similares. O desenvolvimento dessa técnica, aqui chamada de *State Pre-Selection*, será ilustrada considerando um problema de otização estocástica multi-estágio. A técnica proposta será aplicada em um caso baseado no estudo sobre o problema de planejamento energético brasileiro, além disso os resultados serão apresentados e discutidos.

Palavras-Chave: Multi-Stage Stochastic Optimization, Cutting-plane method, SDDP, Cut Selection, Long-term energy planning.

Rio de Janeiro
Janeiro de 2023

Contents

1 Introduction

Nondeterministic long-term operation planning problems have been used in the energy sector for at least 40 years [Bir85; PP85]. In Brazil, this is due to the main energy source used, from the hydroelectric power plants [Sis], a source that relies on stochastic factors, like the water inflow. Also, the interest on these problems comes from the capacity of regulation provided by the large reservoirs in the country. By having these reservoirs Brazil is capable of store energy, in form of water volume, which softens the impact of periods with high demand or low water inflow.

To mathematically model the Energy planning problem we can use *Multistage Stochastic Problem* (MSSP) models. This approach allows us to represent the planning horizon and the uncertainty of the problem. Also, MSSP models are widely used in the energy sector literature: [Gor+92; RG92] use a *Multistage Stochastic Model* to the find the optimal scheduling for the Brazilian and Norwegian Systems respectively; [PD12] where the authors analyse the usage and implementation of coherent risk for the New Zealand system; [Sha+13] a study from 2013 that compares MSSP's, in the Brazlian system, with risk neutral and risk averse approaches.

Multistage stochastic problem models allow us to take decisions for each period. Also, with these models we will have an uncertainty, a set of constraints and an objective function for each period. To represent each period we can also use *Dynamic Programming* to enable us to take the decisions *separately* for each period, for each possible uncertainty.

To solve the energy planning problem with multistage stochastic models those works, as we will do, use the *Stochastic Dual Dynamic Programming algorithm* (SDDP), introduced by Pereira and Pinto in [PP91]. Although there are other approaches like [CS87; Ros81], where the authors opted for a deterministic approach, the SDDP algorithm is capable of dealing with models that use uncertainties without losing much computational tractability. The SDDP algorithm consists on approximating the *cost-to-go functions*, or *Future Cost Function (FCF)* as we will call, by piecewise linear functions. The FCFs are provided by the dynamic programming formulation in the form of optimization sub-problems, built from the last period to the first one. The piecewise linear functions are outer approximations and are the maximum of linear lower bound functions, these functions are called *cuts*.

The SDDP usage in this sector brought with it many studies involving different ways to generalize the algorithm, like generalizing SDDP for problems with integer variables [ZAS19; ACF20]. In [ZAS19] the authors propose building cuts to the convex approximations of integer problems as an approximation for the FCFs, with exact cuts at the binary points. Instead of using standard cuts for a convexification, the authors in [ACF20] propose the usage of nonlinear cuts for MILP problems.

Also, the SDDP usage brought studies in SDDP convergence [PG08] where the authors prove almost sure convergence, in finite number of iterations, of a class of algorithms that includes SDDP, for stochastic multistage linear pro-

gramming problems. This result was then extended for general convex problems in [GLP13], that uses Lipschitz estimates to study the cuts quality, and that is an idea explored in this work.

Besides guaranteeing that the SDDP algorithm will converge in finite number of iterations, we can't guarantee that this number will be small. Having high number of iterations can lead to high computational time and also to ill conditioned problems at later iterations. At later iterations we will have an approximation composed by many cuts. With more cuts the chance of having almost parallel cuts increases and this would lead to the ill conditioned problems.

To deal with the high number of cuts at last iterations most algorithms use *cut selection* methods. These methods propose to reduce the number of cuts that will be active, while SDDP is iterating. The main idea is to make inactive as many useless cuts as possible. A cut is considered to be useless if it do not present any improvement to the approximation. To do the cuts selection there are different approaches as presented in [DPF15; Gui+17].

In [DPF15] we are presented to dominance cut selection, where the algorithm only keeps as active the cuts that are above the other in at least on point. Verifying this condition for every cut is hard, so the authors also present the level-1 cut selection, a heuristics that verifies the dominance only in the states visited. With level-1 we are able to use the cut selection as an online procedure without too much additional computational cost. Also, in [Gui+17] the authors prove the SDDP algorithm convergence with Level-1 cut selection.

Although cut selection can help on reducing the problems of later SDDP iterations, we still have to deal with the high number of iterations of the SDDP algorithm. To deal with this problem we explored the fact that in the energy sector we run similar problems every period. So, we can use the results of these problems that we solve to help solving new problems. So, this work's contribution is to propose a method that aims to speed up SDDP, using the information of previous runs, focused in the energy sector.

The energy planning problem in Brazil is solved every month, day and hour with different models for each one of these periods. But, in each period the problem solved has some similarities with the ones solved at previous periods, or is the same, if dealing with periodic models [SD20]. So, in this work, we focus on this characteristic, of solving similar problems repeatedly, to develop the *State Pre-Selection Algorithm* (SPS).

This algorithm consists in observing states where there were many good cuts built for the FCF approximation of the problems solved in the past. To do that we developed a procedure that we called backward-forward selection. It identifies, using a metric that we developed (LP High Dominance), which cuts from the approximation of the FCF of the past problems presented a high improvement to the approximation. Also, and most important, it gives us in which state there was a high improvement.

With that information we can build cuts in the these locations in order to start solving the problem with a better approximation. In this work we claim that we won't need too many cuts to start with a good approximation. We will

also prove that with L-Lipschitz value functions we can have a upper bound for the amount of cuts needed to have a good approximation.

For our algorithm we developed the way to build the set of candidate states based on the FCF approximations of the problems solved in the past. Also, using a standard machine learning method (K-means), we cluster the collected states to indicate to the SDDP algorithm where to build cuts at the first iterations, when solving new problems. Starting the solution of a new problem with these induced cuts can help the algorithm on warming up, because the SDDP won't start with a blind initial approximation, as it is used. This *warm-start* may speed up the SDDP convergence process, specially with few iterations.

As this algorithm is based only in selecting states to point where to build initial cuts, we think that it can be easily applied to the algorithms proposed in [ZAS19; ACF20]

Finally, we will present results of the State Pre-Selection procedure in a multistage problem, incorporating these modifications into an SDDP implementation in Julia, the package `SDDP.jl` [Dow20]. The models we consider are simplified versions of the Brazilian energy planning problem.

2 Energy Planning Problems and Convex Optimization

In this chapter, we will discuss the energy planning problem so we can contextualize the concepts that we will present. Also, the technique developed in this work is inspired on methods used to model and solve this problem, so understanding the energy planning problem is fundamental to make clear this chapter and the following ones.

Following that, we will walk through the basis to understand convex optimization, heavily inspired on [BV04]. We start from optimization models until we reach convex optimization, when we will see a little of convex analysis. Then we will talk about some properties of convex optimization problems that will be fundamental for the next chapters.

2.1 Energy Planning Problem

The energy planning problem consists of minimizing the cost incurred by electric power plants to supply the demand along a certain period of time, taking in consideration the physical and operational system constraints. The model used in the following sections is inspired from Brazil's energy planning model.

Here we will only consider hydro and thermal power plants as the energy sources. These two power sources are the ones traditionally represented in the Brazilian model, since they account for most energy generation, as can be observed in figure 1, taken from [Sis]. Moreover, hydro power represents the main source of regulation, because water reservoirs absorb the water inflow variation impacts. Also, for simplification, we only consider costs for thermal plants, because the hydro prices are orders of magnitude smaller.

In the following sections, we will present models that can translate this problem into mathematical language, respecting the properties that we will need to use methods such as SDDP.

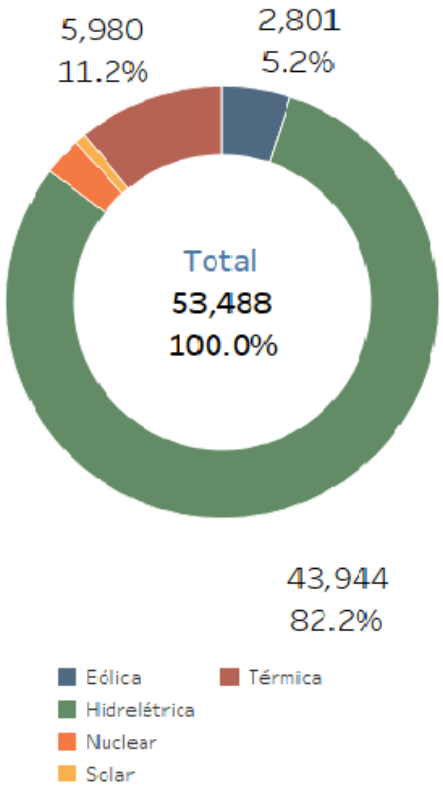


Figure 1: Brazilian power source average composition from 1999 until 2022 - green is the hydro and red is the thermal. Data from ONS [Sis].

2.2 Optimization for general models

In *Mathematical Optimization*, we aim to minimize a function, under certain conditions. This function is called the *objective function*, and its domain dictates the first of many conditions in our model. In equation (1), we show a generic constrained optimization model and the objects that compose it:

$$\begin{aligned} Q = \min_{x} \quad & c(x) \\ \text{s.t.} \quad & g(x) \leq b \\ & h(x) = d \\ & x \in X. \end{aligned} \tag{1}$$

where

- x is the *decision variable*,

- $c(x)$ is the objective function,
- "s.t." means "subjected to",
- X is c's domain,
- $g(x) \leq b$ are the *inequality constraints*,
- $h(x) = d$ are the *equality constraints*,
- and Q is the lowest value that can be achieved by $c(x)$ with $x \in X$ and respecting both inequality and equality constraints.

It is important to notice two things in (1). First, not all problems have a minimum solution, in some cases they can't reach a solution or even this solution is $Q = -\infty$. Because those cases are rare in the energy planning problem scope, we will use minimum, instead of using infimum, as an abuse of notation. Second, in our model, b and d could be incorporated into $g(x)$ and $h(x)$, so that the *right-hand side* (RHS) of the constraints in (1) become simply 0. So, in this work's general models we will omit the RHS variables, unless we need to mention.

In optimization problems we have the concept of the *feasible set*. This is the set of all points that respect the problem's constraints. In (1) these are the points in

$$F = X \cap \{\, x \mid g(x) \leq 0 \,\} \cap \{\, x \mid h(x) = 0 \,\}. \tag{2}$$

If F is the empty set, we say that the problem is *infeasible*: there is no $x \in X$ that respects all constraints simultaneously, so the problem has no solution.

In constrained optimization models, usually, both inequality and equality constraints represent several constraints at once. This means that these are like

$$g(x) = (g_1(x), \ldots, g_n(x)) \preceq (0, \ldots, 0), \tag{3}$$

where $(g_1(x), \ldots, g_n(x)) \preceq (0, \ldots, 0)$ means that $g_i(x) \leq 0$ for all $i \in \{\, 1, \ldots, n \,\}$. The vector form in (3) can be also applied to equality constraints. In both cases, the intention is to summarise all the inequality and equality constraints.

We can translate our energy planning problem to the formulation in (1) through the following model:

$$\begin{aligned} Q(\mathrm{vol}_0) = \min_{\mathrm{vol}} \quad & c^\top \mathrm{thermgen} \\ \text{s.t.} \quad & 0 \leq \mathrm{vol} \leq \mathrm{MaxVol}; \\ & 0 \leq \mathrm{thermgen} \leq \mathrm{MaxTgen}; \\ & 0 \leq \mathrm{hydgen} \leq \mathrm{MaxHgen}; \\ & \mathrm{hydgen}^\top 1 + \mathrm{thermgen}^\top 1 = \mathrm{demand} \\ & \mathrm{vol}_0 = \mathrm{vol} + \mathrm{hydgen} \end{aligned} \tag{4}$$

where

- $\mathbf{vol_0}$ is the initial reservoirs volumes and is a known parameter,
- $Q(\mathbf{vol_0})$ is the problem value,

- **vol** is a decision variable, and represents the water reservoir's volume at the operation's end,
- c is the vector of costs per unit of thermal energy, and represents our objective function in the inner product with **thermgen**,
- **thermgen** is the thermal energy generation vector, where each entry represents a thermal power plant, also a decision variable,
- **hydgen** follows the same **thermgen** logic, but for hydro energy and water reservoirs and is our last decision variable,
- **MaxVol** is the vector representing the maximum volume at each hydro power plant reservoir,
- **MaxTgen** and **MaxHgen** are the maximum energy generation capacity vector for thermal and hydro power plants respectively,
- and finally, **demand** represents the energy demand.

For simplicity, all variables represent energy equivalent units. So, the water stored in reservoirs is not measured in cubic meters, but rather in the corresponding energy that could be generated. This allows us to subtract hydro generation from reservoir volume, for example. Also, in (4) the constraints like

$$0 \leq \text{vol} \leq \text{MaxVol} \tag{5}$$

are a shorthand of the constraints

$$\begin{aligned} \text{vol} &\leq \text{MaxVol}, \\ -\text{vol} &\leq 0. \end{aligned}$$

Finally, observe that, despite minimizing in **vol**, the objective function depends on **thermgen**. This may seems strange, but in (4) equality constraints we have a relation between the thermal energy generation and the reservoirs volume at the operation's end. In other words, if we decide how much water will last in reservoirs, we indirectly decide the amount of thermal energy generated. So, probably, if we chose a value for **vol** we fix the other decision variables value.

2.3 Convex optimization

With the general optimization problem formulation established, we will define the concepts of *convex set* and *convex function*, in order to define *Convex Optimization Problems* and discuss its properties.

Definition 2.1 (Convex Set)**.** *A set $X \subset \mathbb{R}^n$ is said to be convex if, for every x_1 and x_2 in X and $\theta \in [0,1] \subset \mathbb{R}$,*

$$y = \theta x_1 + (1-\theta)x_2 \in X. \tag{6}$$

That means that, if we consider the line segment between any two points of our set X, all points in this segment must be in X for it to be convex.

Definition 2.2 (Convex Function)**.** *A function $f(x) : X \subset \mathbb{R}^n \to \mathbb{R}$, with X a convex set, is said to be convex if, for all x_1 and x_2 in X and $\theta \in [0, 1] \subset \mathbb{R}$*

$$f\big(\theta x_1 + (1-\theta)x_2\big) \leq \theta f(x_1) + (1-\theta) f(x_2). \tag{7}$$

With definitions 2.1 and 2.2 we are now able to define what is a convex optimization problem.

Definition 2.3 (Convex Optimization Problem)**.** *An optimization problem is said to be* convex *if both the objective function and the feasible set are convex. In the context of* (1)*, this means that $c(x)$ is a convex function and that $X \cap \{\, x \mid g(x) \leq 0 \,\} \cap \{\, x \mid h(x) = 0 \,\}$ is a convex set.*

Finding the feasible set could be a problem as hard as the original problem itself. So, to help identifying if the feasible set is convex or not, without visualizing it explicitly, it is an option to check if all the inequality constraints are given by convex functions, and if the equality constraints are given by affine functions. For model (1) that means having $g(x)$ to be convex and $h(x)$ to be an affine function. This implies that modeling our problem with a convex $c(x)$, a convex $g(x)$ and an affine $h(x)$ will give us a convex problem.

Now the question is "What are the advantages of this class of problems?". The first property, to answer this question, is that convex functions, on convex sets, achieve local minimums only if they are a global minimum too. That means that, for convex problems, if an algorithm finds any local minimum for the objective function, in the feasible set, it also solves the problem.

Also, when working with convex optimization problems, a vast set of algorithms, that rely on subgradient properties, becomes usable. To understand why convex optimization is fundamental for this algorithms we need to talk about duality, the subject explored in next section.

2.4 Duality

When working with constrained optimization problems, a way to dodge dealing with a complex constrained feasible set is to try something called *optimization with penalty*. The idea is to remove the constraints and add a penalty to the objective function, charging for getting out of the desired feasible set. It is like charging someone for each unit he consumes beyond a limit previously established. If the penalty price is big enough, this person won't exceed this limit in order to minimize the cost.

For our energy planning model (4), take

$$\text{penalty(hydgen,thermgen)} = \mu(\text{hydgen}^\top 1 + \text{thermgen}^\top 1 - \text{demand}), \tag{8}$$

with $\mu \leq 0$, as an example. By taking out the demand constraint from model (4) and changing the objective function to

$$c^\top \text{thermgen} + \text{penalty(hydgen,thermgen)}, \tag{9}$$

we can now have feasible solutions where the demand is not respected.

However, if $\mathbf{hydgen}^\top 1 + \mathbf{thermgen}^\top 1 < \mathbf{demand}$, we will have

$$\text{penalty(hydgen,thermgen)} > 0. \tag{10}$$

So, if $|\mu|$ is high enough, we can have (9) larger then the operational price inside the constrained set, even if $c^\top \mathbf{thermgen}$ is smaller. On the other hand, if $|\mu|$ is too high, we will stimulate a decision where we generate more energy then the problem demand. It would be more profitable to generate the maximum energy possible at all power plants, even if we don't need it. It's like to receive money for spending energy at home.

As this example shows, finding a good penalty is not a simple task. To have a penalty problem equivalent to the original one we will need to find an optimal penalty function too. So, to build the problem of finding this optimal function we need to define some concepts first. Also, in the following definitions we will use (1) as the reference model.

Definition 2.4 (Lagrangian of an Optimization Problem)**.** *The Lagrangian associated to the optimization problem* (1) *is given by*

$$L(x, \lambda, \mu) = c(x) + \lambda^\top g(x) + \mu^\top h(x), \tag{11}$$

where λ is called the Lagrange multiplier for the inequality constraints and μ the Lagrange multiplier for the equality constraints.

Now, let's define the function that returns the lowest cost by fixing the Lagrange multipliers, that is, the value of the penalized problem:

Definition 2.5 (Lagrangian Dual Function)**.** *The Lagrangian dual function is*

$$d(\lambda, \mu) = \inf_x \quad L(x, \lambda, \mu). \tag{12}$$

It is important to notice that $\lambda \geq 0$ implies in (12) becoming a lower bound to the original problem, for $x \in X$.

Proposition 2.6. *If $\lambda \geq 0$,*

$$d(\lambda, \mu) \leq Q. \tag{13}$$

Proof. To prove (13), first remember that Q is the result of the problem given by (1). Then, let's call $d_r(\lambda, \mu)$ the Lagrangian dual function restricted to the problem's feasible set. And, given that dual functions are an infimum, we have that

$$d(\lambda, \mu) \leq d_r(\lambda, \mu), \tag{14}$$

because, if an infimum is taken in a set that contains another, this infimum is lower or equal to the infimum taken in this second set.

Now, if x is a feasible point, that means it respects the problem's constraints and that implies

- $g(x) \leq 0$ and
- $h(x) = 0$,

that also implies

$$d_r(\lambda, \mu) \leq c(x) + \lambda^\top g(x) + \mu^\top 0. \tag{15}$$

Also, because λ has non-negative entries, we get $\lambda^\top g(x) \leq 0$ and that applied to (15) gives

$$d_r(\lambda, \mu) \leq c(x), \tag{16}$$

for all feasible x.

Finally, joining (14) with (16) we will have

$$\begin{aligned} &d(\lambda, \mu) \leq d_r(\lambda, \mu) \leq c(x) \Rightarrow \\ &d(\lambda, \mu) \leq Q. \end{aligned}$$

The last step comes from the fact that the previous inequality is valid for all feasible x. And this concludes the proof. □

Also, the Lagrange multipliers are like taxes and the problem becomes finding the best values to represent the original problem. This means, for the energy planning problem, finding μ so that it is discouraged to not generate exactly the demand or not having an ending volume that corresponds to the initial one minus the hydro generation. Also, this means to find a λ that discourages exceeding the maximum generation capacity at the power plants.

Now, we can use definition 2.5 to produce an underestimate of the original problem value. Notice that, if we find best lower bound, that is the highest one, and if there are μ and λ where (13) is an equality, we solve the problem of finding the optimal penalty functions. Finding the best lower bound is equal to searching for the pair (λ^*, μ^*) so that

$$d(\lambda^*, \mu^*) \geq d(\lambda, \mu) \text{ for all } \lambda \geq 0 \text{ and } \mu. \tag{17}$$

Looking for the optimal lower bound is one of the basis for algorithms like SDDP, and so it will have a major role on further chapters, especially when we discuss solution methods. To find (λ^*, μ^*), the best Lagrangian multipliers, so we can find the best possible lower bound, we need to solve the so called *dual problem*. This problem comes from maximizing the Lagrangian dual function for the dual variables λ and μ with the constraint $\lambda \geq 0$. Without this constraint, using the previous dual function formulation, it is not guaranteed that (2.5) defines lower bound, because we cannot guarantee (16).

Definition 2.7 (Dual Problem)**.** *The dual problem is given by*

$$\begin{aligned} d^* = \max_{\lambda,\mu} \quad & d(\lambda,\mu) \\ s.t. \quad & \lambda \geq 0. \end{aligned} \tag{18}$$

where d^* is the value of the dual problem, analogous to Q, the value of the primal problem.

It is said that *weak duality* holds because d^*, the dual problem value, is always less than or equal to Q, the optimal value for the primal problem. When working with primal convex problems, with few other conditions, *strong duality* holds: that is, we have $d^* = Q$. One set of conditions that ensure strong duality is called *Slater's conditions* and they demand that there is a point $x \in X$ such that

- $g(x) < 0$,
- $h(x) = 0$.

Notice that the main difference from x being a feasible point is that the inequality constraints are *strict* at this point.

It is important to observe that, under these conditions, solving the dual problem gives us the primal value, as we wanted. In that case, we also say that the dual and primal problems are equivalent.

Now, another topic to discuss on duality is the study of sensitivity analysis when dealing with convex functions. We want to know how much Q varies if we perturb the constraints. That means what happens to Q if we have

- $g(x) \leq \epsilon_g$
- $h(x) = \epsilon_h$

in (1) instead of 0 at the constraints RHS.

In [BV04, chapter 5, section 5.6.3] we have that, if $Q(\epsilon_g, \epsilon_h)$ is differentiable and strong duality holds, then the Lagrangian multipliers λ^* and μ^* equals

$$-\frac{\partial Q(\epsilon_g, \epsilon_h)}{\partial \epsilon_g} \tag{19}$$

and

$$-\frac{\partial Q(\epsilon_g, \epsilon_h)}{\partial \epsilon_h}, \tag{20}$$

respectively, at $(\epsilon_g, \epsilon_h) = (0, 0)$. Here, (λ^*, μ^*) is the dual problem solution, that is the Lagrangian multipliers pair that maximizes the dual problem. This relation will be important later, when we present cutting planes in chapter 4.

Finally, we will talk about the most important result for this work, related to duality.

Proposition 2.8. *As described in [BV04, chapter 5, section 5.6.2], if strong duality holds and the dual optimum is attained we have that*

$$Q(\epsilon_g, \epsilon_h) \geq Q(0,0) - {\lambda^*}^\top \epsilon_g - {\mu^*}^\top \epsilon_h. \tag{21}$$

Proof. Let x be any feasible point, for the perturbed problem, that means the problem with the perturbed constraints

- $g(x) \leq \epsilon_g$,
- $h(x) = \epsilon_h$.

By strong duality,

$$\begin{aligned} Q(0,0) = d(\lambda^*, \mu^*) &\leq c(x) + {\lambda^*}^\top g(x) + {\mu^*}^\top h(x) \\ &\leq c(x) + {\lambda^*}^\top \epsilon_g + {\mu^*}^\top \epsilon_h. \end{aligned}$$

With that we have

$$c(x) \geq Q(0,0) - {\lambda^*}^\top \epsilon_g - {\mu^*}^\top \epsilon_h.$$

As this inequality holds for every feasible x we can conclude (21). □

Observe that (21) gives us a lower bound affine function for $Q(\epsilon_g, \epsilon_h)$, the optimal value function of the perturbed problem. This will be the bases to explain the algorithm explored in chapter 4 that base the SDDP.

3 Multistage Stochastic Optimization Problems

In this chapter I will present a summary of Stochastic Optimization, Multistage Optimization Problems and the Dynamic Programming formulation. These topics are fundamental to understand the MSSP formulation that is used to model the energy planning problem presented in this work. That formulation, in turn, will be the key to understand the following chapters.

3.1 Stochastic Optimization

In the energy planning problem, there are many nondeterministic aspects, such as the energy demand, the inflow at the reservoirs and many others. Here we will focus only on the inflow nondeterministc variables of this model. Although there are approaches by deterministic models [CS87] to deal with this problem, this work will focus on the *stochastic optimization* models.

In our stochastic optimization models for the energy planning, inflow will be modeled as a random variable. With that in mind, it is reasonable to ask the meaning of minimizing a random variable or a function of random variables. The answer to that question is not simple, and so, in this work, we will use a *risk measure* to translate these random variables into real valued variables.

The most common risk measure is the average, yielding the mean value of a random variable:

$$E[\Xi] = \sum_{i=1}^{N} p_i \xi_i, \tag{22}$$

where Ξ is the random variable, ξ_i is a possible realization of Ξ and p_i is the respective probability. That measure is sufficient if the optimization model isn't risk averse, that is, a model that weighs more the outcomes that represent bad events, like low inflow. Also, this measure preserves convexity.

Theorem 3.1. *Let a function $f(x, \Xi)$ be convex in x for every ξ. We will have that $g(x) = E[f(x, \Xi)]$ is a convex function too.*

Proof. Let $f(x, \Xi)$ be a convex function for every outcome ξ and $\theta \in [0, 1] \subset \mathbb{R}$, so by linearity of the expectation:

$$\begin{aligned} g(\theta x + (1-\theta)y) &= E[f(\theta x + (1-\theta)y, \Xi)] \\ &\leq E[\theta f(x, \Xi) + (1-\theta) f(y, \Xi)] \\ &= \theta E[f(x, \Xi)] + (1-\theta) E[f(y, \Xi)] \\ &= \theta g(x) + (1-\theta) g(y), \end{aligned} \tag{23}$$

so, by convex function definition, it is proved that g is a convex function. □

Now that we understand the choice of using the average function as the measure at our stochastic optimization model, it is time to show how we build this kind of model.

Definition 3.2 (Stochastic Optimization Problems)**.** *A problem defined by*

$$\begin{aligned} Q = \min_{x} \quad & E[c(x)] \\ s.t. \quad & g(x) \leq 0; \\ & h(x) = 0; \\ & x \in X, \end{aligned} \tag{24}$$

with $c(x)$, $g(x)$ and $h(x)$ random functions, is called a stochastic optimization problem.

At this work, take solving a problem with random constraints as being to find the optimal solution in the set of points that are feasible for all outcomes of the random variable. In other words, representing this with a deterministic optimization model, it is equal to having a constraint for each possible outcome of the random variables.

For the energy planning problem, this interpretation is to say that the objective is to minimize the average operational cost by applying an optimal policy that has a feasible solution for every possible outcome of the inflow at the reservoirs.

So, we can cast the energy planning problem (4) in a stochastic formulation with (24):

$$\begin{aligned} Q(\text{vol}_0) = \min_{\text{vol}} \quad & E[c^\top \text{thermgen}] \\ \text{s.t.} \quad & 0 \leq \text{vol} \leq \text{MaxVol}; \\ & 0 \leq \text{thermgen} \leq \text{MaxTgen}; \\ & 0 \leq \text{hydgen} \leq \text{MaxHgen}; \\ & \text{hydgen}^\top 1 + \text{thermgen}^\top 1 = \text{demand} \\ & \text{vol}_0 = \text{vol} + \text{hydgen} - \text{inflow}. \end{aligned} \tag{25}$$

where **inflow** is now a random vector representing the water inflow at each reservoir.

Dealing with a simple random variable may seem easy in a first glance: we only added the **inflow** and an average in the objective function to (4). But, in the next sections it will be exposed that dealing with this model for long term problems will demand more techniques and again a different model formulation.

3.2 Multistage Optimization Problems

Now we will present *multistage optimization problems*, that is a class of problems where decisions are taken in different periods of time. In the energy sector, the main objective is to plan a resource management strategy not just taking into consideration the next hour, day, or month, but to set an optimal strategy for the whole planning horizon. Without looking for the future we could have a low cost at the present, but this could also lead us to higher cost for the whole planning horizon.

For that kind of problem it makes sense to have another model formulation where it is possible to take decisions at each *stage* (month, day, hour). In

multistage optimization problems, a stage is a period where we can take a decision. Then our model must have variables to represent decisions taken at each stage, the cost of these decisions and the constraints for what can be decided at each stage. So the decision variable will become a collection of x_t, one for each stage t.

Definition 3.3 (Multistage Optimization model)**.** *A multistage optimization model can be described by*

$$\begin{aligned} Q(x_0) = \min_{x_1,\ldots,x_T} \quad & \textstyle\sum_{t=1}^{T} c_t(x_t) \\ s.t. \quad & g_t(x_t) \leq 0; \\ & h_t(x_t) = 0; \\ & l_t(x_t, x_{t-1}) = \alpha_t; \\ & x_t \in X_t; \\ & t \in \{\, 1, \ldots, T \,\}, \end{aligned} \tag{26}$$

where t is the index referring to the problem's stage.

One particularity in (26) is the addition of the constraint $l_t(x_t, x_{t-1}) = \alpha_t$. This constraint is the *transition rule constraint* and is responsible for representing how the present *state variable* will be related to the previous state variable. This is like the relation in (4) and (25) represented for the constraint that relates vol_0 with vol. Also, in (26) we have the inequality and equality constraints from (4) depending on t. The equations related to g_t and h_t represents, constraints for the decisions taken at the t-th stage.

Also, in (26), we have a *finite horizon model*, that is, a model where the end of planning horizon is known. In this case, the future after T doesn't matter, either because the process ends at time T, or because we neglect the operation cost after T.

Once again, we can describe the energy planning problem with that formulation, but for now without the stochastic parameters.

$$\begin{aligned} Q(\mathrm{vol}_0) = \min_{\mathrm{vol}_1,\ldots,\mathrm{vol}_T} \quad & \sum_{t=1}^{T} (c_t^\top \mathrm{thermgen}_t) \\ \text{s.t.} \quad & 0 \leq \mathrm{vol}_t \leq \mathrm{MaxVol}_t; \\ & 0 \leq \mathrm{thermgen}_t \leq \mathrm{MaxTgen}_t; \\ & 0 \leq \mathrm{hydgen}_t \leq \mathrm{MaxHgen}_t; \\ & \mathrm{hydgen}_t^\top 1 + \mathrm{thermgen}_t^\top 1 = \mathrm{demand}_t; \\ & \mathrm{vol}_{t-1} = \mathrm{vol}_t + \mathrm{hydgen}_t; \\ & t \in \{\, 1, \ldots, T \,\}. \end{aligned} \tag{27}$$

In (27) the transition constraint is represented by

$$\mathrm{vol}_{t-1} = \mathrm{vol}_t + \mathrm{hydgen}_t. \tag{28}$$

Besides the demand and energy prices, for which it is natural to assume fluctuations over time, it is worth mentioning why other variables can depend on time:

- $\mathbf{MaxVol}_t$, $\mathbf{MaxHgen}_t$ and $\mathbf{MaxTgen}_t$ can change between stages because of structural changes on the power plants. Some examples are maintenance, that reduces the operation of a power plant, or a new power plant that starts operating;

- $\mathbf{vol}_t$, $\mathbf{thermgen}_t$ and $\mathbf{hydgen}_t$ are decision variables, so they represent the decisions taken in each period.

Equation (28) tells us that the reservoirs' volumes at the end of stage t depend on the respective volumes at the end of stage $t-1$. In other words, this shows that in each stage the starting reservoirs volumes are the volumes left at the previous stage end.

Also, this relation shows us the importance of not spending all reservoir water in one stage. If we spend too much water in first stage, for example, we will have a low value for $\mathbf{c}_1^\top \mathbf{thermgen}_1$. But, this will imply on having to generate more thermal energy in future stages, because we won't have water at reservoirs to supply the demand with hydro energy. And, as we can see in (27), the second constraint says that each thermal power plant has a maximum capacity. So, when we need more thermal energy we may need to generate it from more expensive power plants.

3.3 Dynamic Programming model

Before finally talking about MSSPs, I will present the *dynamic programming* formulation for multistage problems. This model will consist in separating the whole problem into many simpler sub-problems that can be solved in a backward fashion, resulting in solving the original problem. That means that, instead of having a model like (26) where there are several constraints and a big objective function built by the sum of stage costs, we will have smaller problems chained with each other.

Definition 3.4 (Multistage Optimization Sub-Problem)**.** *The sub-problems associated to optimization problem* (26) *are:*

$$
\begin{aligned}
Q_t(x_{t-1}) = \min_{x_t} \quad & c_t(x_t) + Q_{t+1}(x_t) \\
s.t. \quad & g_t(x_t) \leq 0; \\
& h_t(x_t) = 0; \\
& l_t(x_t, x_{t-1}) = \alpha_t; \\
& x_t \in X_t,
\end{aligned}
\tag{29}
$$

for t between 2 and $T-1$. Here, $Q_{t+1}(x_t)$, in the objective function, is the optimal value function of the sub-problem for $t+1$, and is also called the future cost function *(FCF). This name is due to the fact that, in this formulation, Q_{t+1} is equal to the cost of all future stages. So, by solving the future stages we can solve the t-th stage too.*

Also, we have a sub-problem for $t = 1$

$$\begin{aligned} Q_1(x_0) = \min_{x_1} \quad & c_1(x_1) + Q_2(x_1) \\ s.t. \quad & g_1(x_1) \leq 0; \\ & h_1(x_1) = 0; \\ & l_1(x_1, x_0) = \alpha_1; \\ & x_1 \in X_1. \end{aligned} \tag{30}$$

Here x_0 is given and doesn't depend on any previous decision, unlike x_{t-1} that is given for other stages, but depends on $Q_{t-1}(x_{t-2})$ solution. And, the optimal value for the first stage sub-problem is in fact the optimal value for the original problem.

Finally, the sub-problem for the last stage $t = T$, where there is no future cost, is simpler, since the objective is only $c_T(x_T)$:

$$\begin{aligned} Q_T(x_{T-1}) = \min_{x_T} \quad & c_T(x_T) \\ s.t. \quad & g_T(x_T) \leq 0; \\ & h_T(x_T) = 0; \\ & l_T(x_T, x_{T-1}) = \alpha_T; \\ & x_T \in X_t. \end{aligned} \tag{31}$$

With the dynamic programming formulation, we start solving the last sub-problem, because it does not depend on the solution for any future stage. Also, it is important to notice that each of these sub-problems contains a simpler objective function and they have less constraints in comparison with the complete formulation from equation (26). In fact, the model given by (26) has around T times more constraints then the sub-problems from (29), (30) and (31).

This formulation allow us, for the energy planning problem, to see each sub-problem as deciding the reservoir volume separately. By deciding the volume at the end of the operation horizon, solving the T-th stage sub-problem, we have the last month (day, hour) operation cost as a function of $\mathbf{vol}_{T-1}$. This gives us the $T-1$-th stage future cost function and allows us to solve the $T-1$-th sub-problem. This, once again, will let us solve the previous sub-problem and by repeating this we can have the original problem solution as a function of $\mathbf{vol}_0$.

3.4 Multistage Stochastic Programs

As mentioned previously, combining the multistage and stochastic models is important to arrive at the formulation that will be used in this work. In this section we present the standard formulation of MSSPs and then discuss what comes with it.

Our formulation 3.6 frames the optimization problem in the dynamic programming setting we have just presented. As stated, it (implicitly) assumes *stagewise independence* of the uncertainty, as will be defined in a moment, which is a common hypothesis for the energy planning problem. Assuming this allows

us to have an FCF for each stage as in the dynamic programming model for the deterministic problem. Without stagewise independence, the FCF would also depend on the past realizations of the stochastic variables Ξ_t, and so we would have to construct an FCF per trajectory. Also, having an FCF for each stage, instead of for each trajectory, is responsible for the computational tractability of the SDDP algorithm.

Definition 3.5 (Stagewise Independence)**.** *A stochastic process $\Xi_t, t \in 1, \ldots, T$ is said to be stagewise independent if Ξ_t does not depend on $\xi_1, \ldots, \xi_{t-1}$.*

Definition 3.6 (Stochastic Multistage Model)**.** *For t between 2 and $T-1$ the sub-problem is given by*

$$\begin{aligned} Q_t(x_{t-1}) = \min_{x_t} \quad & c_t(x_t) + E[Q_{t+1}(x_t)] \\ s.t. \quad & g_t(x_t) \leq 0; \\ & h_t(x_t) = 0; \\ & A_t x_t + B_t x_{t-1} = \alpha_t, \\ & x_t \in X_t, \end{aligned} \tag{32}$$

where the matrices A_t and B_t are random matrices and α_t a random vector. This means that all their entries are random variables. Also, as this constraint represents the transition rule, these matrices are called transition matrices.

For $t = 1$, we have a sub-problem defined by

$$\begin{aligned} Q_1(x_0) = \min_{x_1} \quad & c_t(x_1) + E[Q_2(x_1)] \\ s.t. \quad & g_1(x_1) \leq 0; \\ & h_1(x_1) = 0; \\ & A_1 x_1 + B_1 x_0 = \alpha_1 \\ & x_1 \in X_1, \end{aligned} \tag{33}$$

where A_1, B_1 and α_1 are deterministic.

And, for the T-th stage the sub-problem is defined as

$$\begin{aligned} Q_1(x_{T-1}) = \min_{x_T} \quad & c_T(x_T) \\ s.t. \quad & g_T(x_T) \leq 0; \\ & h_T(x_T) = 0; \\ & A_T x_T + B_T x_{T-1} = \alpha_T \\ & x_T \in X_T, \end{aligned} \tag{34}$$

where the main difference again is the absence of the future cost function.

The change for this more specific transition rule form comes from

$$A_t x_t + B_t x_{t-1} = \alpha_t$$

being a standard representation for this constraint in MSSP models for the energy planning problem. Also, the affine form comes from having a convex problem, so the equality constraints are affine.

And, with that, we end the section about optimization MSSPs models for the energy planning problem. The model established in definition (3.6) contains all that we need to start explaining the SDDP algorithm, and this work's idea to improve it.

4 Cutting Plane Algorithms

After describing the energy planning problem and the model formulation that we will use in this work, it is time to present methods to solve this problem. We start with algorithms for simpler models, and work our way up to an algorithm to solve convex multistage stochastic problems (MSSPs).

Let's start with the method called *cutting plane algorithm* (CPA). This method consists on constructing *cuts*, that is, affine functions that are a lower bound for the future cost function, or the resource cost in two-stage problems. Then, we can use a piecewise linear approximation for the FCF, made by the maximum of those cuts, to solve an approximation to the original problem. By doing that, CPA reduces the problem of solving the sub-problems of a MSSP into solving a series of linear problems.

The advantages of doing this comes from the simplicity of representing computationally the FCF's as these approximations. Because, even after decomposing the original problem into sub-problems, the FCFs are not simple to represent computationally. Also, by having a piecewise linear approximation we only have to solve a linear problem instead of our convex optimization problem with a general nonlinear objective function.

To do that, the idea is to start with a known lower bound as the piecewise linear approximation of the FCF. Then, we compute each stage sub-problem solution with that FCF approximation and use the result to build new cuts. These cuts will be used to improve the FCF's piecewise linear approximation. The improvement is given by taking the maximum of the initial lower bound and the new cuts as the new FCF approximation. This process is meant to be repeated to improve the FCF approximation once again.

That process will happen until some convergence certification is achieved. That means achieving a criterion that guarantees that the FCF's approximation is close enough from the actual FCF.

4.1 Two-Stage Problems

In order to explain how to build those cuts, and how to determine convergence for the algorithm, we will start with the CPA for a two-stage deterministic problem. The two-stage problem is the simplest version of multistage optimization problems. It is commonly used to model problems in which a decision is taken before some event happens and then, after this event happens, another decision is taken. The last decision is usually called recourse.

The two-stage problem will be formulated as

$$\begin{aligned} Q_1(x_0) = \min_{x_1} \quad & c_1(x_1) + Q_2(x_1) \\ \text{s.t.} \quad & A_1 x_1 + B_1 x_0 = \alpha_1; \\ & x_1 \in F_1, \end{aligned} \tag{35}$$

with $Q_2(x_1)$ given by,

$$\begin{aligned} Q_2(x_1) = \min_{x_2} \quad & c_2(x_2) \\ \text{s.t.} \quad & A_2x_2 + B_2x_1 = \alpha_2; \\ & x_2 \in F_2. \end{aligned} \tag{36}$$

From now on, to simplify notation, we use F_t to denote the intersection of X_t with all other constraints in the stage t sub-problem, save for the transition constraints, which we keep explicitly.

Now, let $Q_{k,1}(x_0)$ be the problem with the piecewise linear approximation for the FCF at the k-th iteration of the algorithm. That is:

$$\begin{aligned} Q_{k,1}(x_0) = \min_{x_1} \quad & c_1(x_1) + \mathfrak{Q}_{k,2}(x_1) \\ \text{s.t.} \quad & A_1x_1 + B_1x_0 = \alpha_1; \\ & x_1 \in F_1, \end{aligned} \tag{37}$$

where $\mathfrak{Q}_{k,2}(x_1)$ is the maximum of the k cuts generated at the first k iterations

$$\mathfrak{Q}_{k,2}(x_1) = \max_{i \in \{\,0,\ldots,k-1\,\}} C_i(x_1) = \theta_i + \langle \beta_i, x_1 - x_{i,1} \rangle. \tag{38}$$

After each CPA iteration (or at the beginning of each iteration) it is verified if the convergence was achieved. If not, the algorithm computes $x_{k,1}$, the argmin of $Q_{k,1}(x_0)$, then $Q_{k,2}(x_{k,1})$, which, for two-stage problems, is equal to $Q_2(x_{k,1})$. Then, the algorithm picks the Lagragian multiplier, related to

$$A_2x_2 + B_2x_{k,1} = \alpha_2,$$

that as we saw in section 2.4, gives us $\partial Q_{k,2}(x_{k,1})$. After that, it sets $\theta_k = Q_{k,2}(x_{k,1})$, $\beta_k = \partial Q_{k,2}(x_{k,1})$ and so $C_k(x_1) = \theta_k + \langle \beta_k, x_1 - x_{k,1} \rangle$. That is how a cut is constructed in each iteration.

Observe that $C_k(x_1)$ follows a similar shape of (21) that we saw in section 2.4. Here θ_k is the equivalent of the problem value function $Q(0,0)$ in (21) and β_k is the equivalent of the Lagrangian multiplier λ^*. This gives us a lower bound for $Q_{k,2}(x_1)$ and so, as $Q_{k,2}(x_1) = Q_2(x_1)$, we will have a lower bound for $Q_2(x_1)$.

With that, the first stage FCF's approximation becomes

$$Q_{k+1,2}(x_1) = \max_{i \in \{\,0,\ldots,k\,\}} C_i(x_1) = \theta_i + \langle \beta_i, x_1 - x_{i,1} \rangle, \tag{39}$$

and, because $\mathfrak{Q}_{k,2} \leq \mathfrak{Q}_{k+1,2}$, we will have that $Q_{k,1}(x_0) \leq Q_{k+1,1}(x_0)$. So, the approximation sequence is non decreasing with each algorithm iteration. Also, due to the fact that each cut individually is made below Q_2 it is also true that $Q_{k,1}(x_0) \leq Q_1(x_0)$ for every iteration. And so, this shows that CPA gives an under approximation for $Q_1(x_0)$ that is non decreasing.

4.2 Two-Stage Nondeterministic CPA

Now, we will consider the changes that are made to solve two-stage nondeterministic models with CPA. The main idea here will be to use the expectation of

the cut functions, an *average* cut. It is important to remember that we assume a discrete distribution for our random variables.

So, let's describe the two-stage problem stochastic model. Also, now we will identify the random variables indices and inputs that were omitted, for simplicity, in chapter 3.

$$\begin{aligned} Q_1(x_0) = \min_{x_1} \quad & c_1(x_1) + E_{\xi_1}[Q_2(x_1, \xi_1)] \\ \text{s.t.} \quad & A_1 x_1 + B_1 x_0 = \alpha_1, \\ & x_1 \in F_1, \end{aligned} \tag{40}$$

$$\begin{aligned} Q_2(x_1, \xi_1) = \min_{x_2} \quad & c_2(x_2) \\ \text{s.t.} \quad & A_{2,\xi_1} x_2 + B_{2,\xi_1} x_1 = \alpha_{2,\xi_1}, \\ & x_2 \in F_{2,\xi_1}. \end{aligned} \tag{41}$$

Here ξ_1 is the random variable observed at the end of the first stage. You can observe that there is no ξ_0, because we start the problem with that value just established.

For that model, the approximation $Q_{k,1}(x_0)$ will be defined as having $\mathfrak{Q}_{k,2}(x_1)$ determined by the maximum of the average cuts. The average cut done at the i-th iteration is determined by

$$C_i(x_1) = \sum_{\xi_{1,j} \in \Xi_1} p(\xi_{1,j}) \big(\theta_{i,j} + \langle \beta_{i,j}, x_1 - x_{i,1} \rangle\big). \tag{42}$$

Here $\xi_{1,j} \in \Xi_1$ means for every possible outcome of Ξ_1. Also, $\theta_{i,j}$ and $\beta_{i,j}$ are $Q_{i,2}(x_{i,1}, \xi_{1,j})$ and $\partial Q_{i,2}(x_{i,1}, \xi_{1,j})$ respectively. Where $Q_{i,2}(x_{i,1}, \xi_{1,j})$ the objective value function of the second stage with $\xi_{1,j}$ being the observed value at the end of the first stage. At last, $x_{i,1}$ is the argmin of $Q_{1,i}(x_0)$.

So, the model with the piecewise linear approximation for the FCF will be like

$$\begin{aligned} Q_{k,1}(x_0) = \min_{x_1} \quad & \Big(c_1(x_1) + \mathfrak{Q}_{k,2}(x_1) \\ \text{s.t.} \quad & A_1 x_1 + B_1 x_0 = \alpha_1; \\ & x_1 \in F_1, \end{aligned} \tag{43}$$

where

$$\mathfrak{Q}_{k,2}(x_1) = \max_{i \in \{\, 0, \ldots, k-1 \,\}} C_i(x_1), \tag{44}$$

with $C_i(x_1)$ defined as in (42).

4.3 Multistage CPA

Now, we consider the multistage setting. There are no great conceptual changes, but it is necessary to explain how the approximations for the FCF of each stage.

As we did for the two-stage problems, let's start with the model for multistage deterministic problems.

Definition 4.1 (Multistage Deterministic CPA)**.** *The k-th iteration problem is given by*

$$\begin{aligned} Q_{k,T}(x_{T-1}) = \min_{x_T} \quad & c_T(x_T) \\ s.t. \quad & Ax_T + Bx_{T-1} = \alpha_T; \\ & x_T \in F_T, \end{aligned} \tag{45}$$

$$\begin{aligned} Q_{k,t}(x_{t-1}) = \min_{x_t} \quad & c_t(x_t) + \mathfrak{Q}_{k,t+1}(x_t) \\ s.t. \quad & A^\top x_t + B^\top x_{t-1} = \alpha_t; \\ & x_t \in F_t, \end{aligned} \tag{46}$$

$$\mathfrak{Q}_{k,t+1}(x_t) = \max_{i \in 0,\ldots,k-1} C_{i,t}(x_t) = \theta_{i,t} + \langle \beta_{i,t}, x_t - x_{i,t} \rangle. \tag{47}$$

for t = 2,...,T-1, and

$$\begin{aligned} Q_{k,1}(x_0) = \min_{x_1} \quad & c_t(x_1) + \mathfrak{Q}_{k,2}(x_1) \\ s.t. \quad & A^\top x_1 = 0; \\ & x_1 \in F_1. \end{aligned} \tag{48}$$

We can observe from definition 4.1 that we have a FCF approximation for each stage. So, that means that in each iteration the algorithm will generate $T-1$ new cuts, updating all future cost functions. Also, keep in mind that this updates do not happen at the same time, as we will see.

It is time to give a name for the algorithm steps. First, the *forward pass*, where the algorithm verifies the convergence and then, if convergence is not reached, defines the states where the new cuts will be calculated. The *backward pass*, where the algorithm computes each one of the demanded cuts and adds these to the respective stage cuts collection. It is worth noting that here is where the dynamic programming formulation presented earlier starts to shine. It's mainly because of the way that these steps are implemented that this formulation is needed, as it will be clear.

The forward step has this name because it happens from the very first stage to the last one. This is a necessary procedure because we need to compute $x_{k,t-1}$ to solve $Q_{t,k}(x_{k,t-1})$. As pointed out in chapter 3, the $t-1$-th decision variable is the t-th stage incoming state. In the energy planning problem, this corresponds to the need of computing the volume at the end of stage $t-1$, $\mathbf{vol}_{t-1}$. Without $\mathbf{vol}_{t-1}$ we can't solve the t-th stage approximated problem, because we won't know the starting volume, and so how much water we can spend.

On the other hand, the backward pass occurs in the reverse order: It starts at $t = T$ and ends at $t = 2$. After doing the forward pass, we have the point $x_{1,T-1}$ necessary to compute $Q_{1,T}(x_{1,T-1})$ and $\partial Q_{1,T}(x_{1,T-1})$. Like in the two-stage problem, these values are used to create the new cut, following the formula

$$C_{1,T}(x_{T-1}) = \theta_{1,T} + \langle \beta_{1,T}, x_{T-1} - x_{1,T-1} \rangle,$$

with $\theta_{1,T}$ and $\beta_{1,T}$ given by $Q_{1,T}(x_{1,T-1})$ and $\partial Q_{1,T}(x_{1,T-1})$ respectively. Also, this cut will immediately be added to the $T-1$-th stage cuts set and so $\mathfrak{Q}_{1,T}(\cdot)$ is now updated to $\mathfrak{Q}_{2,T}(\cdot)$ and, by definition, $Q_{1,T-1}(\cdot)$ will be updated to $Q_{2,T-1}(\cdot)$. Notice here that $Q_T(\cdot) \equiv 0$, so we can take $\mathfrak{Q}_{i,T+1}(\cdot) = 0$ for every $i \in 1, \ldots, k$, and there will be no updates for $Q_{1,T}$.

After having the updated $Q_{2,T-1}(\cdot)$ and $x_{1,T-2}$, this second obtained from the forward pass, it is possible now to compute

$$C_{1,T-1}(x_{T-2}) = \theta_{1,T-1} + \langle \beta_{1,T-1}, x_{T-2} - x_{1,T-2} \rangle.$$

Similarly, $\theta_{1,T-1}$ and $\beta_{1,T-1}$ are given by $Q_{2,T-1}(x_{1,T-2})$ and $\partial Q_{2,T-1}(x_{1,T-2})$, which means that $C_{1,T-1}(\cdot)$ is constructed with the just updated approximation. That is one reason for doing this step from stage T to 2.

The other stages will follow this same procedure, which ends at the second stage because at the first one there is not a previous future cost function approximation to be updated. With that step finished the first algorithm iteration is now ended and so it will go to the next one beginning from the forward pass again.

Another topic that will be important to understand the next chapter is the fact that, unlike the two-stage problem, the FCFs of $Q_{k,t}(\cdot)$, for $t \in \{\, 2, \ldots, T-2 \,\}$, are not the exact future cost. This will only happen at $t = T$. At the multistage problems the cuts are a lower bound for $Q_{k,t}(\cdot)$ that is a lower approximation for the original objective value function. So, these cuts are a lower bound affine function for a lower approximation. For $t = T-1$ the cuts are tight, but this is not true for the cuts constructed with $t \in \{\, 2, \ldots, T-2 \,\}$.

Despite that, while the algorithm runs the iterations, all FCFs are getting better piecewise linear approximations, so the cuts computed are getting closer to the actual function. Besides that, it is necessary to observe that the last stages will have a better approximation. The approximation for the last stage FCF is the actual last stage cost, as in the two-stage problem. So the cuts to approximate the second last stage sub-problem are made tangent to the FCF. But, to approximate the third last sub-problem, the cuts are made tangent to $Q_{k,T-1}$ instead of Q_{T-1}. So the cuts are not guaranteed to be tight to the actual FCF. This means that after each iteration the FCFs for the first sub-problems tends to be not as well approximated as for the last ones.

4.4 Multistage Nondeterministic CPA

For the multistage nondeterministic models, we will have to do the same changes that we did for two-stage nondeterministic problem. Once again we will have average cuts. But the average cuts are computed in a slightly different way. Also, the algorithm that we will describe here is finally the SDDP.

Definition 4.2 (Multistage Stochastic Optimization CPA)**.** *The CPA multi-*

stage stochastic optimization model at the k-th iteration is given by

$$\begin{aligned} Q_{k,T}(x_{T-1}, \xi_{T-1}) = \min_{x_T} \quad & c_T(x_T) \\ s.t. \quad & A_{T,\xi_{T-1}} x_T + B_{T,\xi_{T-1}} x_{T-1} = \alpha_{T,\xi_{T-1}}; \\ & x_T \in F_{T,\xi_{T-1}}, \end{aligned} \tag{49}$$

for the last stage;

$$\begin{aligned} Q_{k,t}(x_{t-1}, \xi_{t-1}) = \min_{x_t} \quad & c_t(x_t) + \mathfrak{Q}_{k,t+1}(x_t) \\ s.t. \quad & A_{t,\xi_{t-1}} x_t + B_{t,\xi_{t-1}} x_{t-1} = \alpha_{t,\xi_{t-1}}; \\ & x_t \in F_{t,\xi_{t-1}}, \end{aligned} \tag{50}$$

for t between 2 and $T-1$. And, for the first stage, we have

$$\begin{aligned} Q_{k,1}(x_0) = \min_{x_1} \quad & c_1(x_1) + \mathfrak{Q}_{k,2}(x_1) \\ s.t. \quad & A_1 x_1 + B_1 x_0 = \alpha_1; \\ & x_1 \in F_1. \end{aligned} \tag{51}$$

Here

$$\mathfrak{Q}_{k,t+1}(x_t) = \max_{0 \le i \le k-1} C_{i,t}(x_t), \tag{52}$$

for t between 1 and $T-1$. Where $C_{i,t}(x_t)$ is the cut built at the i-th iteration for the t-th stage.

Similarly to the two-stage problem, we define the average cut by

$$C_{i,t}(x_t) = \sum_{\xi_{t,j} \in \Xi_t} p(\xi_{t,j}) \big(\theta_{i,t,j} + \langle \beta_{i,t,j}, x_t - x_{i,t} \rangle \big). \tag{53}$$

Here $\theta_{i,t,j}$ and $\beta_{i,t,j}$ have almost the same interpretation as in the two-stage problem, with the difference that we have one for each stage t. The big difference is $x_{i,t}$ that now is the argmin of $Q_{t,i}(x_{i,t-1}, \xi_{i,t-1})$. Notice that we need to fix ξ_{t-1} to solve the approximated t-th stage objective value function.

At MSSPs, for every iteration i, we fix a trajectory $(\xi_{i,1}, \xi_{i,2}, \ldots, \xi_{i,T-1})$, at the forward, so we can compute $(x_{i,2}, x_{i,3}, \ldots, x_{i,T-1})$. For MSSPs it is important to have a high amount of iterations, so we can visit many trajectories. Also, notice that the cuts that will be generated will visit the states related to these trajectories. That means that $x_{i,t}$'s given at the forward are related to the sampled $\xi_{i,t}$'s. That makes this algorithm a stochastic algorithm.

It is worth mentioning that the "S" from SDDP comes for this reason. It is because SDDP is a stochastic algorithm, not because it is used to deal with stochastic optimization problems.

4.4.1 SDDP Convergence Stop Criterion

The convergence of $Q_{k,1}(x_0)$ to $Q_1(x_0)$ does not always occur under a reasonable number of iterations. So, in order to find a stopping criterion that is computationally tractable, algorithms that use cutting plane algorithms can use also an

inner approximation. Then, these algorithms measure how close both approximations are (to imply convergence) as described at [PMF13]. For two-stage problems that is the usual criterion.

Other than that there is the usage of an statistical approximation to estimate the confidence interval as shown in [PP91]. This confidence interval cannot be to large, otherwise there could be a chance of being too far from convergence.

Also, one could set a limit for the number of iterations. When using an iteration limit we take in consideration the fact that there are some convergences guarantees, for a large number of iterations. These guarantees imply in a small gap between the piecewise linear approximation, $\mathfrak{Q}_{k,t}(x_{t-1})$, and the real future cost $Q_t(x_{t-1})$.

In general, what will happen in SDDP is the usage of a combination of these techniques. For example, the algorithm will stop either with a certified convergence by the inner-outer approximation gap, or if it reaches the N-th iteration.

By now, we have established the basic concepts in the formulation of SDDP algorithm. The Stochastic Dual Dynamic Programming algorithm is a way to use the cutting planes obtained with Lagrangian multipliers to get a subproblem approximation, given by the dynamic programming formulation, to solve the original problem. And, to deal with the stochastic aspects of the model, we rely on the stagewise independence property that allow us to have a FCF for each stage instead of one FCF per trajectory.

5 Cut Selection

As discussed in chapter 4, the method used to solve the energy planning problem involves approximating the FCF by a piecewise linear lower bound function. In order to improve that approximation, at each iteration the algorithm computes average cuts and adds them to each sub-problem approximated FCF. To compute the variables that compose a cut it is necessary to solve a sub-problem approximation, that is computing $Q_{k,t-1}(x_{t-1})$ given by (52).

In general, the FCF approximation, $\mathfrak{Q}_{k,t}(x_{t-1})$ in (52), can be represented as

$$\begin{aligned} \mathfrak{Q}_{k,t}(x_{t-1}) = \min_{x_{t-1}} \quad & \Theta \\ \text{s.t.} \quad & \Theta \geq C_{1,t}(x_{t-1}); \\ & \vdots \\ & \Theta \geq C_{k-1,t}(x_{t-1}), \end{aligned} \tag{54}$$

where $C_{i,t}$ for $i \in \{\, 1, \ldots, k-1 \,\}$ are the average cuts obtained at the SDDP iterations. This formulation makes explicit the fact that adding a cut corresponds to adding a new constraint to a Linear Problem (LP).

By having a high amount of constraints in a LP, the time needed to solve it will grow. Also, solving a LP with lots of constraints may lead to ill-conditioned matrices and therefore amplify numerical errors. So in order to efficiently perform more iterations, increasing the solution precision, it is necessary to avoid those problems that comes having from a large number of cuts.

One approach is to verify redundancy between the cuts in order to take out the unnecessary ones added during the whole process. The idea is to reduce the amount of cuts without reducing the problem precision. These unnecessary cuts may exist because during algorithm early to middle iterations some improvement for the piecewise linear approximation can be superseded by the addition of a later cut. As it was said in chapter 4, the earlier cuts are estimated on looser approximations, so they can be too far from the actual FCF to the point of being mostly under the latter cuts.

For example, let $\mathfrak{Q}^j_{k,t}(x_{t-1})$, given by

$$\begin{aligned} \mathfrak{Q}^j_{k,t}(x_{t-1}) = \min_{x_{t-1}} \quad & \Theta \\ \text{s.t.} \quad & \Theta \geq C_{1,t}(x_{t-1}); \\ & \vdots \\ & \Theta \geq C_{j-1,t}(x_{t-1}); \\ & \Theta \geq C_{j+1,t}(x_{t-1}); \\ & \vdots \\ & \Theta \geq C_{k-1,t}(x_{t-1}), \end{aligned} \tag{55}$$

for any $j \in \{\, 1, \ldots, k-1 \,\}$, be the piecewise linear approximation at the k-th iteration, for the stage t, but without the j-th cut ($C_{j,t}(\cdot)$). If, in this case, it is true that $\mathfrak{Q}^j_{k,t}(x_{t-1}) = \mathfrak{Q}_{k,t}(x_{t-1}), \forall x_{t-1} \in X$ we say that the j-th cut isn't

useful. This happens because the addition of that constraint to the problem $\mathfrak{Q}_{k,t}^{j}(x_{t-1})$ does not represent any improvement to the FCF approximation.

5.1 Cut Selection Methods

Cut selection methods are algorithms that identify, along the iterations of an algorithms like SDDP, which of the previously generated cuts will be used in the next iterations. They can be distinguished by the frequency in which there will occur a selection between the cuts, if they care about the cut usefulness and how much they care. Also, at most cases, they will store the cuts that will not be used at the approximation, to help on convergence process.

We will describe in what follows some of the most popular methods, along with the positive sides and disadvantages of using each one in comparison with the others.

5.1.1 Last Cuts Method

Last cuts is based on the idea that the cuts generated at the last iterations of SDDP are made on better approximations, so these tend to be closer to the actual FCFs. This method is one of the simplest selection algorithms. It depends on a constant H that refers to the number of most recently added cuts that will be selected to compose the approximations. That means having

$$\begin{aligned} \mathfrak{Q}_{k,t}^{[1,k-(H+1)]}(x_{t-1}) = \min_{x_{t-1}} \quad & \Theta \\ \text{s.t.} \quad & \Theta \geq C_{k-H,t}(x_{t-1}); \\ & \vdots \\ & \Theta \geq C_{k-1,t}(x_{t-1}), \end{aligned} \tag{56}$$

as the piecewise linear approximation.

Although the idea may seem promising, the last cuts approach does not perform well. First, it is not a simple to find the right H. Choosing a small H can be insufficient for some stages approximation, for example. Furthermore, excluding the earlier cuts can imply on an impact at the algorithm convergence, because in most cases there are some useful cuts among them.

There are some possible improvements for this strategy as mentioned in [DPF15], but they won't be discussed here. In fact, this strategy will not be used in this work and only serves as an an example of what can be done with low effort.

5.2 Dominance Cut Selection Methods

The next algorithms are a more sophisticated methods and take into consideration the concept of usefulness of a cut.

5.2.1 Exact Dominance Cut Selection

At *exact dominance cut selection* we keep all cuts that aren't useless. We say that a cut is useless if there are no feasible point where the value of this coincides with the value of the FCF approximation, using all cuts. These cuts are considered to be an inactive ones, and and that is why we won't use these at the FCF approximation composition.

Doing this is equivalent to verifying the following condition

$$\exists x_{t-1} \in X \mid \mathfrak{Q}_{k,t}(x_{t-1}) = C_{j,t}(x_{t-1}). \tag{57}$$

Notice that condition (57) is not equivalent to being useful. In fact, the previous equation can be true at the same time that $\mathfrak{Q}^j_{k,t}(x_{t-1}) = \mathfrak{Q}_{k,t}(x_{t-1})$ for all $x \in X$. In this case, at each x'_{t-1} such that $C_{j,k}(x'_{t-1})$ is the maximum cut, there is at least one other $i \neq j \in 1, \ldots, k-1$ such that $C_{i,k}(x'_{t-1}) = C_{j,k}(x'_{t-1})$.

This method solves last cuts method problem of taking out useful cut. But, despite having a good selection criteria cuts, dominance cut selection method relies on verifying (57). Verify this condition along the SDDP iterations can slow the whole process and not compensate the method usage for some cases.

5.2.2 Level-1 Dominance Cut Selection

In order to find an alternative that can be applied multiple times we have the *level-1 dominance cut selection*. This algorithm is a heuristic, for the previous one, that proposes to check the dominance (57) only on states where we construct the cuts. That means verifying condition (57) only for points $x_{i,t}$, instead of verifying in the whole feasible space.

To do this, let $X_{k,t}$ be the set of all $x_{i,t}$ generated until the k-th iteration for the stage t. The level-1 dominance algorithm selects the cuts $C_{i,t}(\cdot)$ with $i \in I_{k,t} = \bigcup\limits_{x_{j,t} \in X_{k,t}} I^{(x_{j,t})}_{k,t}$, where

$$I^{(x_{j,t})}_{k,t} = \{\, i \in \{\, 1, \ldots, k-1 \,\} \mid C_{i,t}(x_{j,t}) = \mathfrak{Q}_{k,t}(x_{j,t}) \,\}. \tag{58}$$

So, $I_{k,t}$ is the set of cut indices generated until the k-th iteration, for the t-th stage, where the value of this cut at some state visited is equal to the value of FCF approximation at the same point.

It is valid to observe that, for each point $x_{i,t} \in X_{k,t}$, the set $I^{(x_{i,t})}_{k,t}$ can have more then one index. This can lead to a new strategy where, similar to latest cuts method, we keep only the H newer cut indices from $I^{(x_{i,t})}_{k,t}$. Also, it can be made by selecting the H older, as seen in [Gui+17].

The level-1 dominance cut selection receives this name because it keep only the cuts that achieve the highest value at each point from $X_{k,t}$. A similar method, called "Level-2" cut selection, also keep the cuts with second highest value, trying to mitigate the losses in the regions far from $X_{k,t}$.

We can observe at figure 2 the impact of adding the second highest cuts at the states visited. With Level-1 we would only select the orange and red cuts, by

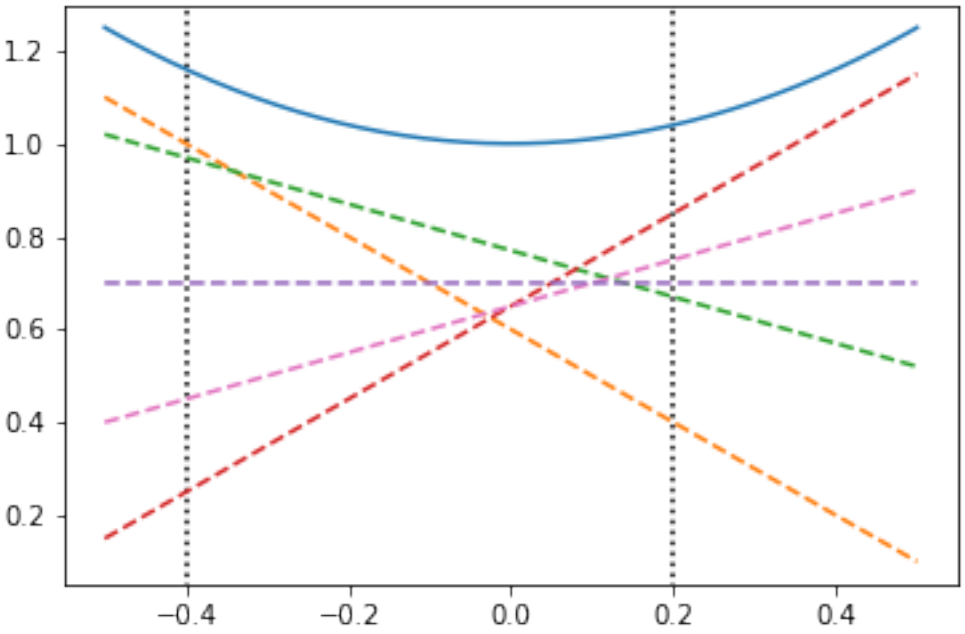


Figure 2: Blue curve is the FCF, the dotted colored lines are the cuts and the dotted black vertical lines represent the states visited.

adding the second highest cuts we would have also the green and pink cuts. By having the green cut we improve the approximation significantly in the states near 0. Also, this idea can be extend to any level-n, $n \in \mathbb{N}$, by following the same logic presented in level-1 and level-2.

5.3 LP Dominance Cut Selection

Another way to select is to keep only the useful cuts, by taking out both the useless and the not useless that are inactive. We developed a way to verify if a

cut is useful by solving

$$
\begin{aligned}
P_{k,t} = \max_{x,\delta,\Theta} \quad & \delta \\
\text{s.t.} \quad & \Theta \geq C_{1,t}(x); \\
& \vdots \\
& \Theta = C_{k,t}(x) - \delta; \\
& \vdots \\
& \Theta \geq C_{i,t}(x); \\
& \delta \geq 0 \\
& x \in X,
\end{aligned} \tag{59}
$$

where i is the iteration and k is the index of the cut analyzed.

At (59) the variable δ is an auxiliary variable that measures how much we can low $C_{k,t}(x)$ until getting useless. There are three possible results for the δ:

- no possible δ, implies that $C_{k,t}(x)$ is useless. This cut is below the other ones;
- $\delta = 0$, implies that $C_{k,t}(x)$ is neither below or above the other cuts. Although it is not below, it is indeed not a useful cut, it is inactive;
- $\delta > 0$, implies that $C_{k,t}(x)$ is useful and δ is also the height above all other cuts. So, in other words, δ measures $C_{k,t}(x)$ dominance, or usefulness.

So, for this method we will keep the cuts where $\delta > 0$. That is solving the PL in (59) for every cuts and using in the FCF approximation only the ones with positive δ.

Solving an LP multiple times can be easier than checking (57). But, is also a procedure that present a increasing high computational cost if we do it multiple times.

5.3.1 LP High Dominance Cut Selection

At last there is *LP high dominance cut selection* criterion, to categorize a cut as an useful one, that we have developed. This criteria is a version of the LP dominance method that cares about how big is the dominance of a cut. Instead of just solving (59) and checking if $\delta > 0$, we verify how big δ is. The idea is to keep only the cuts that present a improvement significant enough by checking if δ is bigger then a pre-established tolerance.

In summary, the *LP High Dominance Cut Selection* verifies how high is the dominance, instead of only verifying if the cut is dominant. Despite being more computational costly, as we also said earlier, we don't intend to use this method multiple times. This method won't be used while SDDP is running, but after solving the SDDP as we will see in the next chapter.

6 State Selection

At this point, we have gone through several concepts that will be used this chapter, so let's do a quick recap. To setup this chapter, we saw several optimization models, walking through convex, stochastic and multistage optimization. Along these topics, we have discussed about duality, dynamic programming model and stagewise independence. Then, I have presented the cutting planes algorithm and how to apply it to MSSPs in the SDDP form. In the applications side, and close to the topic of this chapter, we have discussed in the beginning of this work the Brazilian energy planning problem and how to fit it to our model. Finally, in chapter 5, we have studied the cut selection methods and also exposed a new one that inspired this chapter.

Now, I will develop the State Pre-Selection algorithm for rolling horizon SDDP models that we presented earlier. First, I will explain some characteristics of the real process that lead to this new strategy. Once again, my focus will be on the energy planning problem example. In the sequence, I will go into the details of this algorithm.

Let's remember the idea behind the energy planning problem. The first important aspect is that this problem takes into consideration a whole planning horizon that can be discretized in monthly, daily, or hourly stages. Each of these operational models differs on few aspects from each other, but, most of the times, they have in common the process of re-running them along the whole time horizon as the time passes.

For the energy planning problem, the objective is to use a *rolling horizon* approach. That is, finding a decision policy that minimizes the mean cost along the whole process. But, as the time passes and we observe the realization of the stochastic variables, we can take decisions based on the values of these realizations.

Take, as an example, the monthly planning, for a five years horizon planning. If we solve this problem starting in January we would know only the inflow value until January. But, as we reach February we would have the knowledge of this month inflow realization too. So, instead of using the previous decision policy we compute the problem once again, but with the new statistics collected. Also, instead of using the same horizon we shift the problem, so it computes five years from February, not from the past January. That way, the *rolling horizon* has the same length of five years.

Keeping the same horizon length is important, because in finite horizon MSSPs planning for more years means you have to be more cautious. At the energy planning problem this translates as preserving more water, by supplying the demand with more thermal energy, implying in higher costs. Preserving more water prevents us from activating the more expensive thermal power plants in months with low inflow. And rare events, such as consecutive low inflow months, are more likely to occur when you are planning for a longer horizon.

This works proposes to identify some patterns from data collected during previous runs, and then develop a warm-start procedure to the SDDP algorithm. More precisely, we add some information at the start of the CPA process, that

should lead to faster convergence. As a simple example, suppose that an initial lower bound better than $C_{0,t}(x_{t-1}) = 0$ is known for each stage. By using better approximations, it is reasonable to expect that the convergence will be achieved faster, because the first cuts will have been made with better piecewise linear approximation for the FCFs.

6.1 State Patterns

Let's take again, as the *original problem* (the problem from where we extract the data), the monthly example, starting in January and with a five years horizon:

$$\begin{aligned} Q_1(x_0) = \min_{x_1} \quad & c_1(x_1) + E_{\xi_1}[Q_2(x_1, \xi_1)] \\ \text{s.t.} \quad & A_1 x_1 + B_1 x_0 = \alpha_1; \\ & x_1 \in F_1, \end{aligned} \tag{60}$$

$$\begin{aligned} Q_t(x_{t-1}, \xi_{t-1}) = \min_{x_t} \quad & c_t(x_t) + E_{\xi_t}[Q_{t+1}(x_t, \xi_t)] \\ \text{s.t.} \quad & A_{t,\xi_{t-1}} x_t + B_{t,\xi_{t-1}} x_{t-1} = \alpha_{t,\xi_{t-1}}; \\ & x_t \in F_{t,\xi_{t-1}}, \end{aligned} \tag{61}$$

for $t \in \{\, 2, \ldots, 59 \,\}$ and with

$$\begin{aligned} Q_{60}(x_{59}, \xi_{59}) = \min_{x_{60}} \quad & c_{60}(x_{60}) \\ \text{s.t.} \quad & A_{60,\xi_{59}} x_{60} + B_{60,\xi_{59}} x_{59} = \alpha_{60,\xi_{59}}; \\ & x_{60} \in F_{60,\xi_{59}}. \end{aligned} \tag{62}$$

After solving this problem, all generated cuts are saved and we have the sets $S_t = \{\, C_{1,t}(\cdot), \ldots, C_{K,t}(\cdot) \,\}$ with K equals the number of cuts at stage t. Those cuts could be useful on solving the *shifted problem* by giving a better start approximation, if there is some similarities on both problems.

Notice that by the shifted problem we mean solving the model where the first month is now February and the horizon ends on January of the sixth year after the beginning of the operation. So, this will lead to solving a slightly different problem where the future cost functions at each stage are similar to the respective month at the original problem. From here, I will refer to $Q_t^{(i)}(\cdot)$ as the t-th stage FCF from the model beginning at the i-th month. In other words, $Q_1^{(1)}(\cdot)$ is the first stage FCF of the model beginning in January, whereas $Q_1^{(2)}(\cdot)$ is the first stage FCF of the model beginning in February, analogous (but not equal) to $Q_2^{(1)}(\cdot)$.

Therefore, the question here is whether we can use some cuts from S_t to form a good initial piecewise linear approximation for $Q_t^{(2)}(\cdot)$. At first look this seems reasonable, but in fact using cuts could lead to a wrong approximation. Because even if they are similar, the cuts generated can't be shared, otherwise you would not have any guarantees that they are lower bounds. We only can use the cuts from S_t if we guarantee that no parameter changes between the original and the shifted problem.

But, between solving the original and the shifted problem, in the real world, some parameters can change. One example is the thermal generation capacity, where a new thermal power plant can start operating earlier then the predicted. If that happens we can have a lower cost and so the cuts in S_t, the ones from the original problem, can be not viable cuts. So, these non-programmatic parameter changes could make impossible to ensure that the future cost functions, for the shifted problem, are larger than the corresponding ones in the original problem.

It is important to notice that we are considering uncertainties that we can't incorporate in the model as random variables. These events can happen but aren't reasonable to be considered for the model. The best we can do is to expect these events and don't assume that the FCF won't be lower.

In summary, if we use some cuts from S_t to build a better initial approximation for the shifted problem and if some non-programmatic change happens we could have a initial approximation that is higher then the FCF. So, we look instead for information in the cuts that could help creating some starter cuts. Let's first remember the elements to build a cut $C_{k,t}$:

- $Q_{k,t}(\cdot)$,
- $\partial Q_{k,t}(\cdot)$,
- $x_{k,t}$,

where, again, the first two are respectively the expected FCF approximation and its subgradients, at the k-th iteration for the t-stage. Of those three elements, the last one does not depend on FCF values, because the first two depend on $\mathfrak{Q}_{k,t}(\cdot)$.

With that in mind, the idea is to use the state, $x_{k,t}$, where the cut was built to give us an information for a warm-start. So, we propose to use a few state points to create the first cuts, instead of walking through the normal first forward steps. Because some steps we use were inspired on cut selection, as we explain shortly, this method will be called *State Pre-Selection* (SPS).

6.2 State Pre-Selection

6.2.1 Lipschitz Approximation

In this work we claim that to do this initial better approximation we won't need to many cuts. That means we need to find only few states where we will construct these stater cuts. But, to have a good start approximation, or warm-start, with few cuts we need to guarantee that those cuts were built in states that well distributed in the feasible set.

That isn't a new idea and was also used in [GLP13] to provide bounds for the largest possible distance between a cutting-plane approximation and the true value function of a problem, based on the distance between states where cuts were made. In particular, for problems with compact state space, by selecting a set of states such that their neighborhoods cover the state space, the value

functions will be "well approximated", with a quality depending on the size of such neighborhoods.

Nonetheless, the set of states might be very large in that setting. For this problem, we can have at least an upper bound on the number of states needed, depending on ϵ (how much we want to approximate the value function). This result is similar to corollary 1 of [ZS22]. There, the authors bound the number of iterations needed for convergence by the number of cuts performed, which are in turn bounded by the number of states visited.

Theorem 6.1. *For a single stage problem, let the objective value function, $Q : [-R, R]^d \subset \mathbb{R}^d \to \mathbb{R}$, be L-Lipschitz. If for every point y in the domain there is a point where a cut was made which is is less than $(\frac{\delta}{2})\sqrt{d}$ away, we have that*

$$Q(y) - \mathfrak{Q}(y) \leq L\delta\sqrt{d} =: \epsilon.$$

Also, we can achieve this result with $(\frac{2R}{\delta})^d$ cuts.

Proof. First, because $Q(\cdot)$ is L-Lipschitz, we have that

$$|Q(x) - Q(y)| \leq L|x - y|.$$

Also, any cut $C(\cdot)$ is tangent to $Q(\cdot)$ at the point x where it was made. So, again, because $Q(\cdot)$ is L-Lipschitz, the cut itself is L-Lipschitz, so

$$|C(x) - C(y)| \leq L|x - y|.$$

If the cut was made at x, then $Q(x) = C(x)$, so

$$|Q(y) - C(y)| \leq 2L|x - y|$$

So, if we guarantee that the maximum distance of any point in the grid to the nearest point where a cut was made is less then $(\frac{\delta}{2})\sqrt{d}$ we have that:

$$|Q(y) - C^*(y)| \leq L\delta\sqrt{d}.$$

Where $C^*(y)$ is the cut made at x^*, the point nearest to y where a cut was made.

By definition $\mathfrak{Q}(y) \geq C^*(y)$, so

$$|Q(y) - \mathfrak{Q}(y)| \leq L\delta\sqrt{d},$$

as we wanted for the first part.

Now, if we take points in a grid with spacing δ, then the diagonal is of length $(\delta)\sqrt{d}$. Then, the maximum distance for $y \in [-R, R]^d$ to the nearest point in the grid is

$$dist(y, x^*) \leq (\frac{\delta}{2})\sqrt{d}.$$

Then, if we take cuts in the points in this grid, we will have $(\frac{2R}{\delta})^d$ cuts, which satisfy the approximation for $Q(\cdot)$. □

This theorem is fundamental to for the State Pre-Selection because guarantees that we won't need to use the whole set of states. So, we can build a warm-start with at least a more reasonable amount of cuts.

6.2.2 SPS Cut Selection

In SPS, the selected cuts can be employed to either finding states where useful cuts were built or states where these cuts represent the greater improvement to the approximation. The cut selection in SPS won't serve to filter the cuts to reduce complexity and numerical errors, like it is usually used, as we described in chapter 5.

It is very important to remark that the cut selection here is an offline procedure. We won't use the cuts from the original problem to build the warm-start, we only select these cuts to find the states where they were built or presented high dominance. This selection happens after solving the original problem, so when we have the final approximation for the FCFs for the original problem.

This selection is procedure to filter the cuts that we will consider to find the states. Without this *offline selection* we would have to consider the whole cut set and this would give us a large states set. And as we claimed in the previous section we do not need a large amount of states to have a good approximation.

So, first, to identify if the k-th cut from stage t is useful or not we use the LP high dominance cut selection, so we solve the LP described in (59). But, this time, the cuts used are the generated at the original problem as in

$$\begin{aligned} P_{k,t}^{(1)} = \max_{x} \quad & \delta \\ \text{s.t.} \quad & \Theta \geq C_{1,t}^{(1)}(x); \\ & \vdots \\ & \Theta = C_{k,t}^{(1)}(x) - \delta; \\ & \vdots \\ & \Theta \geq C_{K,t}^{(1)}(x); \\ & \delta \geq 0 \\ & x \in X. \end{aligned} \tag{63}$$

where K is the number of cuts generated. After identifying if $C_{k,t}^{(1)}(x)$ is useful, by verifying if

$$\delta > \text{tol},$$

we save $x_{k,t}^{(1)}$, in the set $l^{(1)}$ mentioned at the start of section 6.2.

Also, at same time, we save $x^{\left(P_{k,t}^{(1)}\right)}$, the solution of (63) in the set $l^{\left(P_{k,t}^{(1)}\right)}$, that is equivalent to $l^{(1)}$ for this other state. This state represents where this cut "helped" the most for the future cost approximation, and so where there was need for improvement. That is the state where the cut presents the highest dominance. By having both $l^{(1)}$ and $l^{\left(P_{k,t}^{(1)}\right)}$, we can chose to use either the states where the cuts were built or the states where they present the highest dominance. For now we will focus only on $l^{\left(P_{k,t}^{(1)}\right)}$, but keep in mind that everything we do with this set we can do using $l^{(1)}$.

Another observation is that finding the right tolerance (**tol**) is indeed not a simple task. If we choose *tol* large enough (this depends on each problem), this

would lead to finding almost no cut; on the other hand, if $tol \approx 0$, we would have to deal with too many cuts.

Proposition 6.2 (*Tol* Limits)**.** *We claim that:*

1. *If* $tol \to \infty$*, so #cuts selected goes to* 0*.*
2. *If* $tol \to 0$*, we will select all useful cuts, even the ones that represent almost no improvement. And that can represent too many cuts.*

6.2.3 Clustering by K-Means

In order to deal with a possible high amount of state candidates, we opted to invest in some algorithm that can extract the information given by these inputs and return us few states capable to condense this information. This filter is necessary because otherwise the algorithm can select many states in the same region. Building nearby cuts is not a problem in itself, but as we saw earlier, if our set of states are well distributed we can reduce the necessity for more cuts. Also, this deal with the impact of choosing low *tol* in the previous step.

With that in mind, we used a clustering method called K-means. Clustering methods are a class of categorization algorithms used in order to split up a data set in groups of nearby points, also known as clusters. In this context, K-means is a method based on linear separations that identifies the clusters by the average of the points inside them.

Let's say that $Y = \{ y_1, y_2, \ldots, y_N \} \subset \mathbb{R}^n$ and we want to find K clusters, $\mathcal{G} = (\mathcal{G}_1, \ldots, \mathcal{G}_K)$, that separate well these points. First, we pass K possible starting clusters means, let's say $o = (o_1, \ldots, o_K)$. Then, the algorithm will perform the following steps:

- Step 1: For i in { 1, ..., K }, identify the possible K clusters as

$$\mathcal{G}_i = \big\{ y \in Y \mid ||y - o_i|| \leq ||y - o_{j \neq i}||;\ j \in \{ 1, \ldots, K \} \big\}.$$

- Step 2: Compute $o_i^* = \frac{1}{|\mathcal{G}_i|} \sum_{y \in \mathcal{G}_i} y$ and save them into $o^* = (o_1^*, \ldots, o_K^*)$.

- Step 3: Verify if $o_i^* = o_i$ for every i in { 1,..., K }. If the equality is true, the algorithm stops here and then $\mathcal{G}_{i \in \{1, \ldots, K\}}$ are the clusters with $o = (o_1, \ldots, o_K)$ being the clusters means. Else, set $o_i = o_i^*$ and go to step 1.

At the end of the algorithm, we get the set of clusters with the respective central points. It is worth mentioning that there are other K-means variants suggesting other metrics to determine if a points belongs to a cluster [SYR13]. Also, there are researchers studying ways of determining a good number of clusters while running K-means [HE03].

In this work we did not use the clusters, instead we focus on the cluster centers, the mean point that identifies each group. This happens because these points summarises the regions of interest and so they can be used as a candidate

for state selection. Also, these centers are guaranteed to be feasible, because they are convex combination of feasible points, because the states selected and passed to the K-means are feasible.

6.3 State Pre-Selection algorithm

We have established everything we needed to describe the algorithm of State Pre-Selection. First, remember that we intend to solve a shifted problem, that is a problem starting periods after the original problem been solved. So, we can collect all cuts generated for the original problem, knowing from which stage they came and then save this information at a data set. That data set will be called $\mathbb{D}^{(\tau)}$, where τ is the stage where the original problem began.

Furthermore, it is important to understand that $\mathbb{D}^{(\tau)}$ is a set of sets. In fact, $\mathbb{D}^{(\tau)} = \left\{ D_t^{(\tau)} \forall t \in \{ 1, 2, \ldots, T \} \right\}$, with T being the number of stages, t referring to the stage and so $D_t^{(\tau)}$ is the subset of cuts taken from this problem's t-th stage. With that we are able to have the states selection in order to solve the new problem.

Here we will select the states that represent the the highest dominance of a cut. That is where the cut present highest improvement to the FCF approximation. These states are the argmax of the problem given by (59).

It is important to notice that in some cases the LP High Dominance Selection algorithm can select no cuts. For example, let's say that the original problem only has parallel cuts, each one distancing from the ones immediately above and below by at most $\frac{tol}{2}$, where tol is the significance of improvement needed. In this case, the value of (59) will be never high enough and no cut will be useful for this metric. This case is illustrated in figure 3.

But, for this example, clearly we want to keep only the highest cut, as in figure 4, because the other ones are useless. For other problems, finding a solution is not so easy. Take an example where we have a 4 cuts that do not yield high improvement, but all of them are useful and by taking out one of them the approximation don't get too worse. But, at this same situation, by taking two of them out, instead of just one, if this is not correctly done, the improvement turns to be significantly lower. This example is shown in figures 5, 6 and 7, where we can see the differences of the maximum distance, or gap, between the FCF and the approximation changing with different cuts selected.

To avoid this problem, we propose a "*backward-forward selection*" mechanism. That is, we create a new $D_t^{(\tau)}$ where first we add cuts by checking its dominance and then there will be another filter to check the useful ones. This new set will be called $\mathbb{D}_f^{\tau} = \left\{ D_{t_f}^{(\tau)} \forall t \in \{ 1, 2, \ldots, T \} \right\}$ and they will represent the final selected cuts.

The "backward-forward selection" is performed independently for each stage $t \in \{1, 2, \ldots, T\}$:

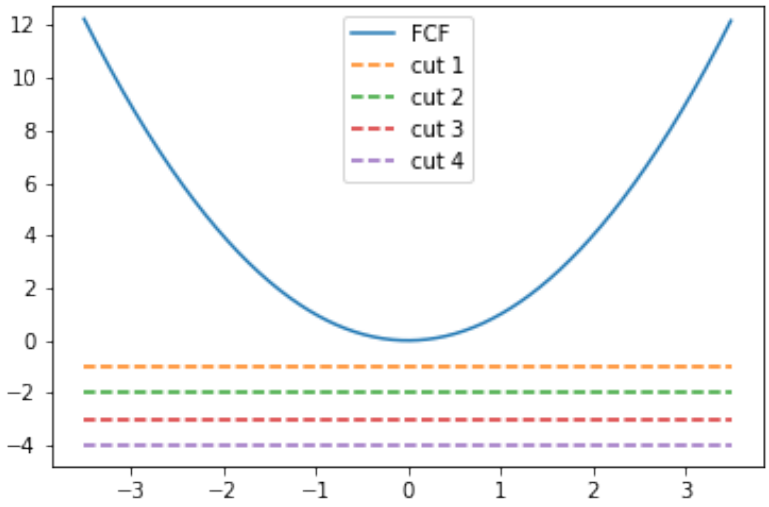


Figure 3: Parallel close cuts before filtration.

Algorithm 1: Backward-Forward Selection

Backward Step:
Add the cut $C_{K,t}$ to $D_{t_f}^{(\tau)}$ and delete it from $D_t^{(\tau)}$;
for $k \in \{K-1\ldots,2,1\}$ **do**
  if $C_{k,t}$ *results in an improvement for larger than tol* **then**
    Include in $D_{t_f}^{(\tau)}$;
  end
end
Forward Step:
for *cut, in the order of inclusion, in* $D_{t_f}^{(\tau)}$ **do**
  if *the cut still does not improve more than tol* **then**
    Remove it from $D_{t_f}^{(\tau)}$;
  end
end

The backward step is responsible for guaranteeing that the cuts represent at least an improvement larger than the tolerance. In this step we guarantee that we will have at least one cut, by starting with latest generated cut and adding only more cuts if they present high dominance at some point. This is important to deal with situations, like in figures 3 and 5, where no cut presents high dominance. Also, the cuts are verified in the opposite order than they were added, because by doing this we verify the better cuts first. In the other hand,

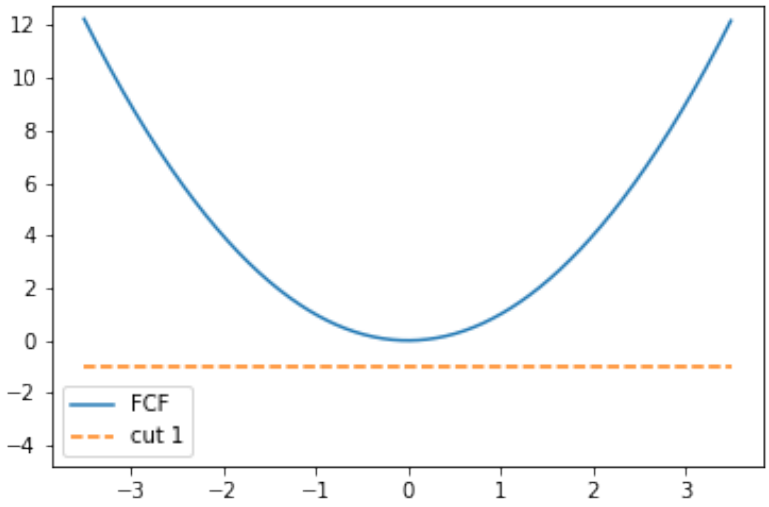


Figure 4: Parallel close cuts after filtration.

the forward step is where we verify if, in the previous step, any of the added cuts became not so useful. Notice that the last cut added at backward step will be maintained, so at the end of the forward step we will maintain at least one cut.

It's important to notice that we save states at the forward step. By doing that these states represent the place where the cuts have high dominance for the original problem FCF approximation using only the selected cuts. The set that represents the states selected at each stage is

$$\mathbb{X}^{(\tau)} = \left\{ \mathbb{X}_i^{(\tau)} \forall i \in \{ 1, 2, \ldots, T \} \right\}. \tag{64}$$

Then, $\mathbb{X}^{(\tau)}$ is used to build the clusters for each stage. By passing a predefined number of clusters, let's say K, we cluster $\mathbb{X}_i^{(\tau)}$ for every $i \in \{ 1, 2, \ldots, T \}$, with K-means. This will return $\mathcal{G}_i^{(\tau)} = \{ \mathcal{G}_{1,i}^{(\tau)}, \ldots, \mathcal{G}_{K,i}^{(\tau)} \}$. These states are guaranteed to be feasible, and so eligible to be used at the warm-start, because they are convex combinations of feasible points of a convex problem.

To do the warm-start, these sets are passed to the SDDP. The algorithm will start by building K warm-start cuts, for each stage, that are cuts at the states selected. After that, we run SDDP normally until convergence or the iteration limit is reached.

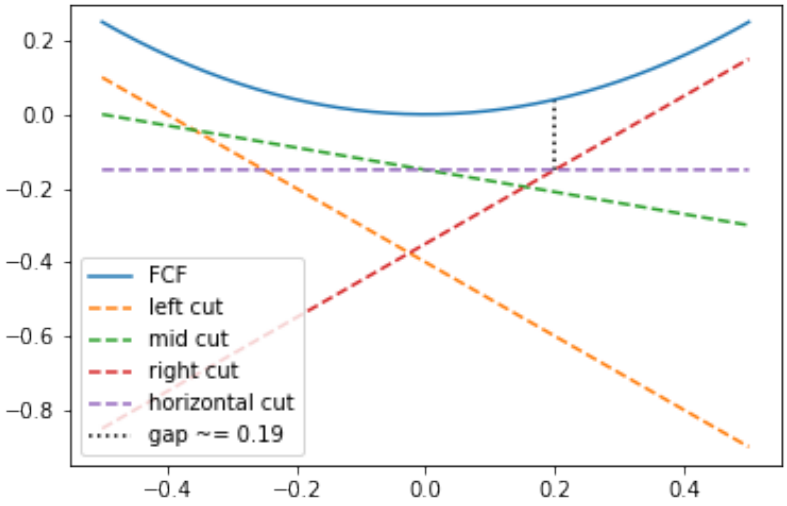


Figure 5: Complex case without Cut Selection.

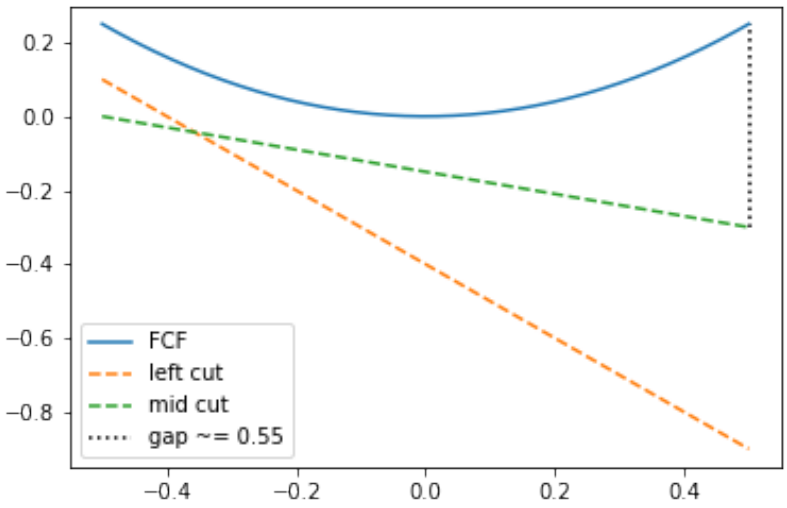


Figure 6: Complex case with a bad Cut Selection filter.

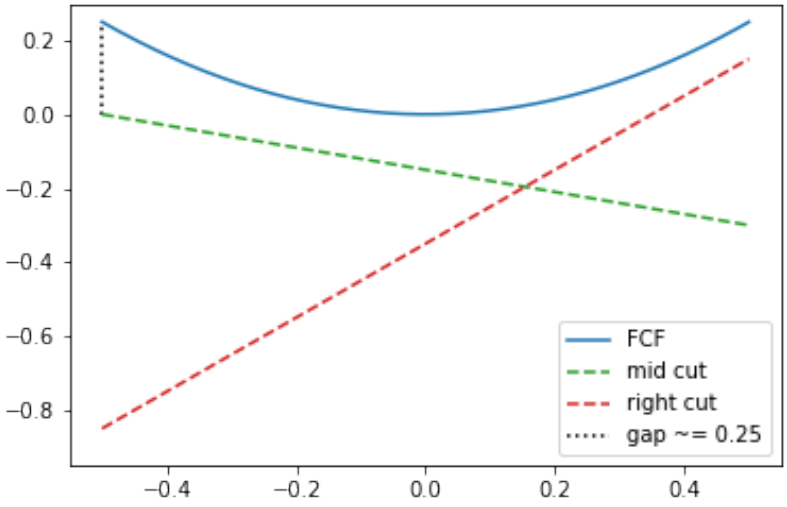


Figure 7: Complex case with a good Cut Selection filter.

6.3.1 Numerical Example

To illustrate the algorithm numerically we did a simple numerical example with a 1D toy energy planning model. The model was given by the equation:

$$
\begin{aligned}
Q_t(\text{vol}_{t-1}) = \min_{\text{vol}_t} \quad & c^\top \text{thermgen}_t + E_{\xi_t}[Q_{t+1}(\text{vol}_t, \xi_t)] \\
\text{s.t.} \quad & 0 \leq \text{vol}_t \leq \text{MaxVol}; \\
& 0 \leq \text{thermgen}_t \leq \text{MaxTgen}; \\
& 0 \leq \text{hydgen}_t \leq \text{MaxHgen}; \\
& \text{hydgen}_t^\top \mathbb{1} + \text{thermgen}_t^\top \mathbb{1} = \text{demand} \\
& \text{vol}_{t-1} = \text{vol}_t + \text{hydgen}_t.
\end{aligned} \tag{65}
$$

For this example we had:

- 5 stages.
- 2 thermal power plants with maximum capacity of 6 and costs 2 and 50 per unit of energy.
- 1 hydro power plants with reservoir maximum volume equals 12, maximum generation capacity equals 12 and initial volume equals 12.
- Also, the demand in each stage is equal to 12.

For this example we have the inflow distribution given by $\{1, 2, 3, 4, 5\}$ at even stages and $\{0, 1, 2, 3, 4\}$ at odd stages. Also, we let SDDP only run 10 iterations, so we had only 10 cuts per stage. By fixing the seed in (11235) we obtained the approximated FCF, at the fourth stage , given the the following cuts.

1. (θ = 212.0, π = -50.0, x = 0.0)
2. (θ = 35.7, π = -21.2, x = 4.2)
3. (θ = 31.2, π = -21.2, x = 4.5)
4. (θ = 71.6, π = -40.4, x = 3.0)
5. (θ = 112.0, π = -50.0, x = 2.0)
6. (θ = 112.0, π = -50.0, x = 2.0)
7. (θ = 112.0, π = -50.0, x = 2.0)
8. (θ = 112.0, π = -50.0, x = 2.0)
9. (θ = 112.0, π = -50.0, x = 2.0)
10. (θ = 112.0, π = -50.0, x = 2.0)

Here, π is the angle of the cut, θ is the height of the cut and x is the state where this cut was computed. Also, we choose the fourth stage because it is easier to visualize with it.

The *state selection starts here*. After having an approximation for this problem, if we want to speed up the the process of solving a problem similar to (65) we need to first find the states candidates for the warm-start. So we used the backward-forward selection on these 10 cuts, with a tolerance of $5e-5$.

With that we obtained that the only cuts that presented a high dominance was

- (θ = 31.2, π = -21.2, x = 4.5),
- (θ = 71.6, π = -40.4, x = 3.0).

Also, the states where these presented the high dominance were respectively at volume equals 12.0 and 3.0.

With the states selected we now cluster these using K-means. As this is a simple example the algorithm will return, by choosing 1 cluster only, the state 7.5. We use 1 cluster only, because that was the average amount of dominant cuts per stage and we used this average to indicate us the number of clusters in this work. That means, we observe the amount of cuts selected in each stage, we sum this amounts and divide by the number of stages. That ends the state selection.

This procedure is done for every stage with the same tolerance. At the end we will have one state for every stage. So, when we would solve the similar problem we start with a cut in those states for every stage. That means each stage FCF will start with an approximation with one cut.

7 Numerical Experiments

In this chapter we present the results obtained by implementing the SPS method on SDDP.jl to solve some problems. Those problems are simple representations of the actual model used to solve the Brazilian energy planning problem. Despite having simpler models, the numerical experiments can show how the application of this method impacts on results and gives us a glimpse of what can be achieved by applying it to more complex models.

But, before describing the problems and presenting the numerical results, let's talk about the computational tools used at this work. The programming language used was Julia, mostly because of the `SDDP.jl` package [DK17], which provides a framework for modeling multistage stochastic optimization. Besides, we use JuMP [DHL17], which provides a syntax to describe general mathematical optimization problems that is mirrored to the used in literature.

Other then that, we used the GLPK package that allows the usage of the Linear programming solver with the same name, which is needed to solve each stage sub-problem, in order to build the cuts. For the State Pre-Selection part involving K-means, we used the Clustering package. Finally, there are still LinearAlgebra for numerical purposes, Random for the statistical part and Matplotlib [Hun07] to build the plots used for graphic visualisation in this work.

7.1 Shifted Models

As a first case study for the State Pre-Selection algorithm, we used the comparison between two shifted problems. In one of them, we solved with the vanilla SDDP, and the other one was solved with the State Pre-Selection modification. We also compared the results of using random states to build the warm-start cuts: instead of running normally or using State Pre-Selection states to build the first cuts, we use random selected states. This third case is a control measure to separate improvements that would not be related to the method proposed in this work.

The original problem, in which the states are collected and the time reference to the shifting, is given by the dynamic programming recursion

$$
\begin{aligned}
Q_1(\text{vol}_0) = \min_{\text{vol}_1} \quad & c^\top \text{thermgen}_1 + E_{\xi_1}[Q_2(\text{vol}_1, \xi_1)] \\
\text{s.t.} \quad & 0 \leq \text{vol}_1 \leq \text{MaxVol}; \\
& 0 \leq \text{thermgen}_1 \leq \text{MaxTgen}; \\
& 0 \leq \text{hydgen}_1 \leq \text{MaxHgen}; \\
& \text{hydgen}_1^\top 1 + \text{thermgen}_1^\top 1 = \text{demand} \\
& \text{vol}_0 = \text{vol}_1 + \text{hydgen}_1;
\end{aligned}
\tag{66}
$$

where vol_0 is given,

$$\begin{aligned} Q_t(\text{vol}_{t-1}, \xi_{t-1}) = \min_{\text{vol}_t} \quad & c^\top \text{thermgen}_t + E_{\xi_t}[Q_{t+1}(\text{vol}_t, \xi_t)] \\ \text{s.t.} \quad & 0 \leq \text{vol}_t \leq \text{MaxVol}; \\ & 0 \leq \text{thermgen}_t \leq \text{MaxTgen}; \\ & 0 \leq \text{hydgen}_t \leq \text{MaxHgen}; \\ & \text{hydgen}_t^\top 1 + \text{thermgen}_t^\top 1 = \text{demand} \\ & \text{vol}_{t-1} = \text{vol}_t + \text{hydgen}_t - \xi_{t-1}; \end{aligned} \tag{67}$$

for $t \in \{ 2, \ldots, T-1 \}$ and with

$$\begin{aligned} Q_T(\text{vol}_{T-1}, \xi_{T-1}) = \min_{\text{vol}_T} \quad & c^\top \text{thermgen}_T \\ \text{s.t.} \quad & 0 \leq \text{vol}_T \leq \text{MaxVol}; \\ & 0 \leq \text{thermgen}_T \leq \text{MaxTgen}; \\ & 0 \leq \text{hydgen}_T \leq \text{MaxHgen}; \\ & \text{hydgen}_T^\top 1 + \text{thermgen}_T^\top 1 = \text{demand} \\ & \text{vol}_{T-1} = \text{vol}_T + \text{hydgen}_T - \xi_{T-1}. \end{aligned} \tag{68}$$

In (66), (67) and (68) :

- vol_t is the state variable and is the vector corresponding to the stored volume at the end of stage t.
- ξ_t is the water inflow at reservoirs at the end of stage t.
- c is thermal power generation cost vector, with each entry corresponding to the respective thermal power plant. For this problem prices won't change between stages.
- $thermgen_t$ is the energy generated by thermal power plants at the t-th stage and is also a vector corresponding to each thermal power plant generation.
- $hydgen_t$ is as above, but for hydro energy.
- $MaxVol$ is the vector representing maximum volume at each water reservoir and that won't change between stages.
- $MaxTgen$ represents the maximum thermal energy generation capacity vector which is the same for every stage.
- $MaxHgen$ follows the same logic as the previous one but for hydro energy.
- And *demand* is the total energy that must be generated.

To collect results and make comparisons with the state pre-selection performance we have solved this model for different values of those variables. This was made to mitigate having biased results, that is having the proposed algorithm performance depending strongly on a particular set of parameters.

With that said, the first results we will present correspond to the case where the number of hydro power plants and thermal power plants are 2 and 3 respectively, we have $T = 20$ stages, and in all of them we have:

- $c = (2, 5, 10)$;
- $MaxTgen = (6, 6, 6)$;
- $MaxHgen = (3, 4)$;
- $MaxVol = (6, 8)$;
- $vol_0 = (3, 4)$;
- $demand = 10$;

with c entries with units of money / power, while the others are simply in energy units.

Also, ξ_t (the inflow) is a 2 dimensional vector where each entry follows a uniform distribution $[1, 5]$ for the odd months and $[0, 4]$ for the even months. That kind of distribution, depending on t, helps us on analysing the results of the shifted problem. This happens because the only fixed parameter that depends on t in each sub-problem is the ξ_t distribution, so by shifting it we can represent the whole shifted problem. So, we don't have to build another model just to represent the shifted problem.

With that, after solving the original problem, also called *the present model*, with 400 cuts, we collect the useful cuts using the LP high dominance selection. To recall, this method, instead of taking every useful cut, filters only the cuts that deliver an improvement larger than some tolerance, that in this case was fixed at $tol = 5e - 3$. This tolerance value was chosen after some tests in order to filter only few cuts but at the same time having a significant amount.

As we can see, in figure 8 that with $tol = 5e - 1$ we have only few states selected, that means that only few cuts were selected. With this low amount of cuts we could we have, rounded, 1 cut selected in average per stage. It is important to mention that there are many states selected at $(0, 0)$, that is why we only see 5 states and a kernel near the origin in figure 8. So, for a higher tolerance the average number cuts per stage isn't a good metric to determine the number of clusters.

In contrast, in figure 9, where $tol = 5e-5$, we can observe that the number of cuts selected is much bigger, by looking to number of state selected and cluster kernels. Although having a high number of cuts selected isn't intrinsically bad, in this work we are interested in the impact of adding just few warm-start cuts. Also, by lowering the tolerance this much, the code script was running much more slower. Finally, in figure 10 we see the result of choosing $tol = 5e - 3$, there isn't as many cuts selected as with $tol = 5e - 5$. But, in this case, there is more than just one, as we saw with $tol = 5e - 1$.

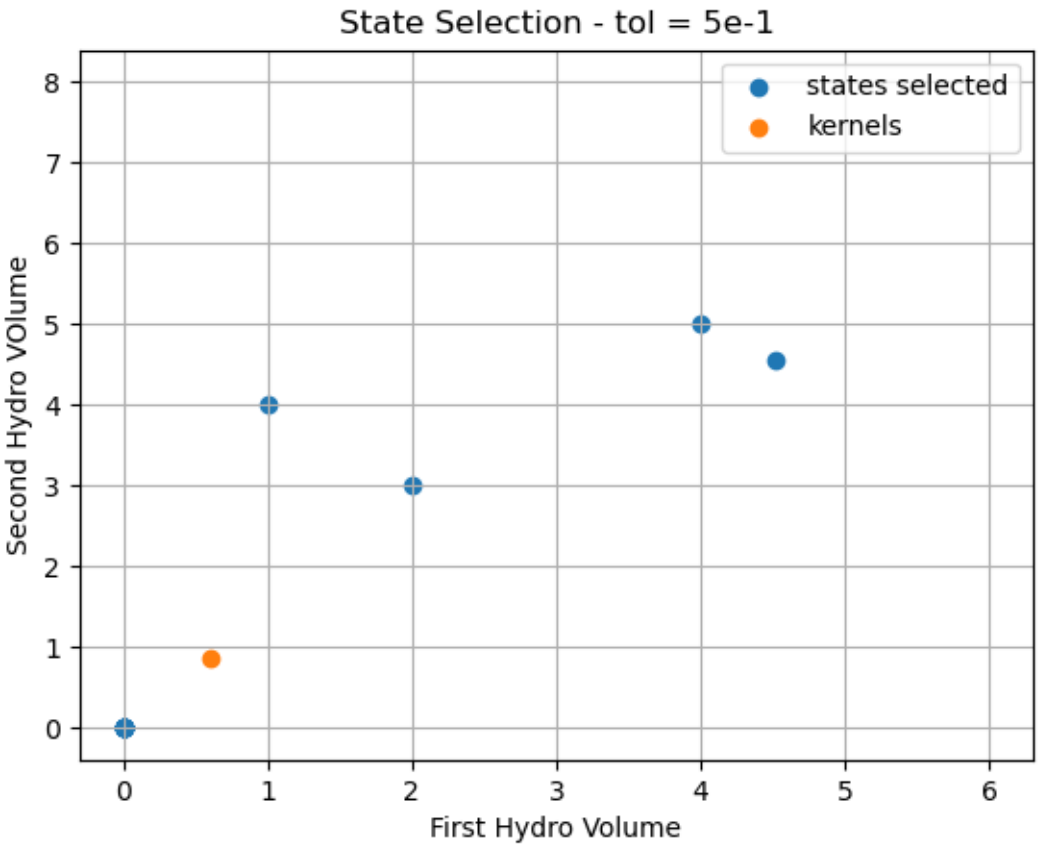


Figure 8: States selected in blue and cluster kernels in orange, with tolerance 0.5.

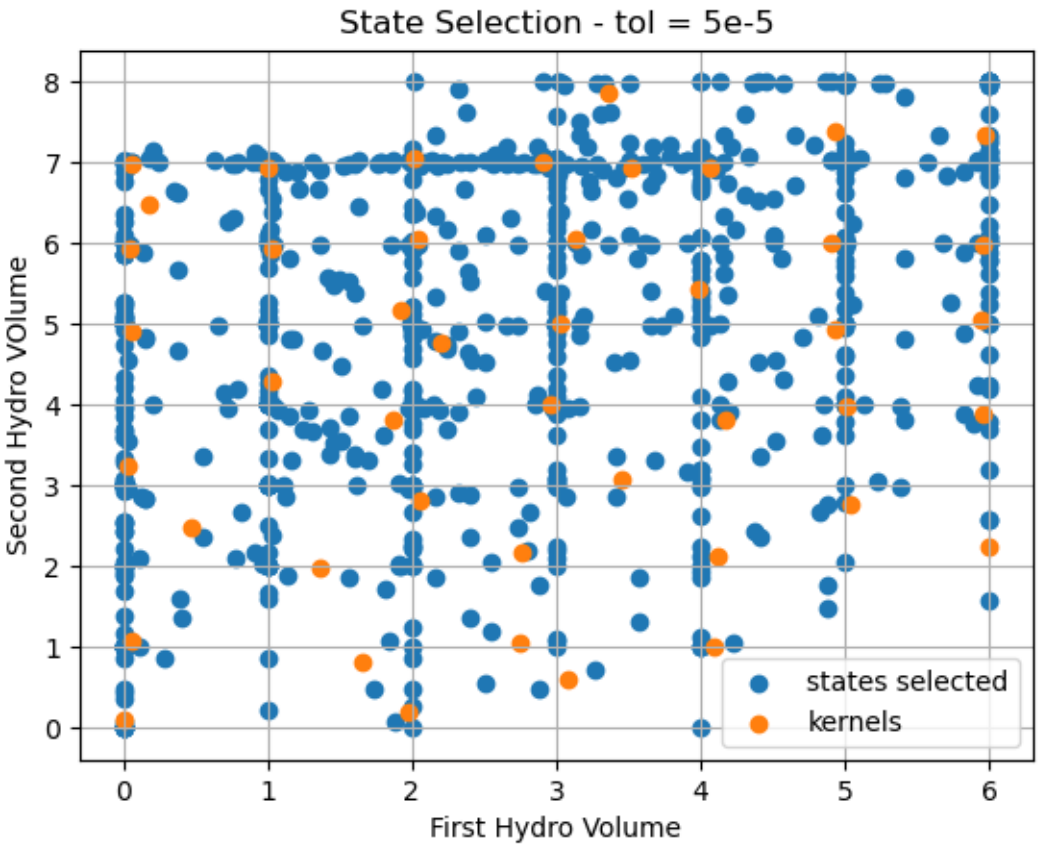


Figure 9: States selected in blue and cluster kernels in orange, with tolerance 0.00005.

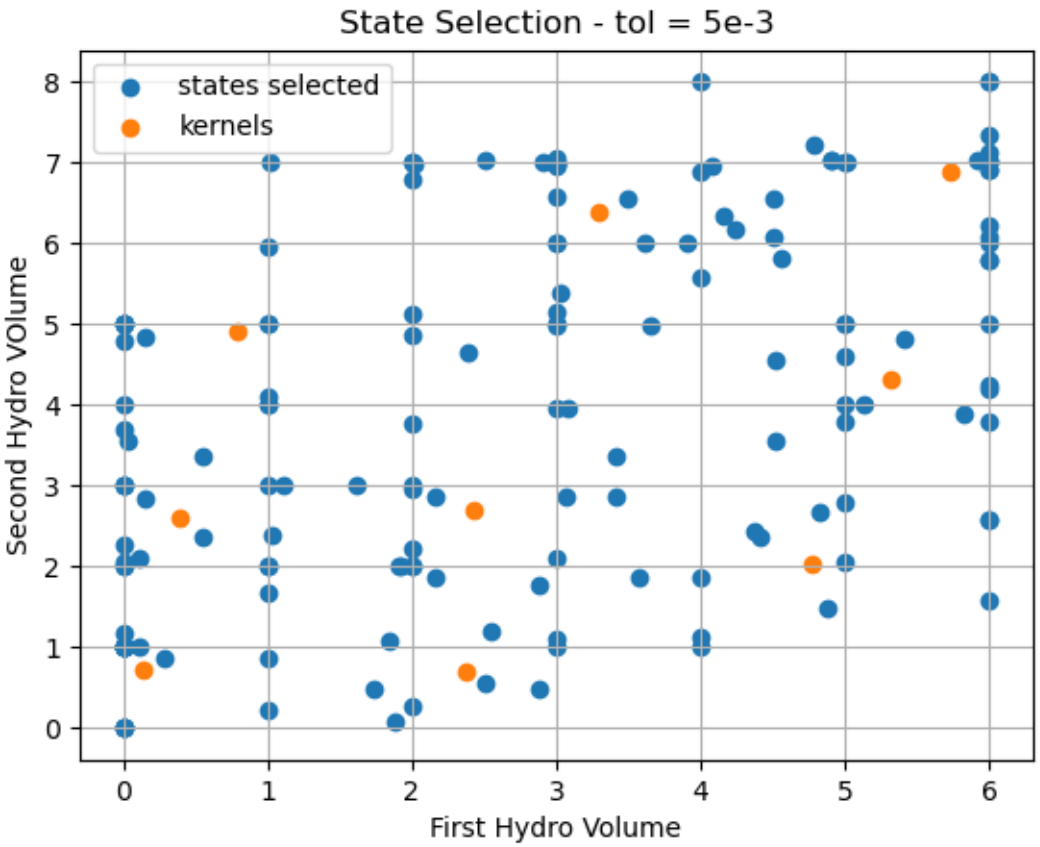


Figure 10: States selected in blue and cluster kernels in orange, with tolerance 0.005.

During the cut selection we compute the states that are the argmax of (59), also we calculate the average amount states selected per stage. Then, because we have similar sub-problems in each stage, we opted for doing the clustering using the states collected for all stages at the same time. Also, the number of clusters is the average amount of states per stage, that for this problem was 9. Using the average amount we guarantee that we do not have too many states at stages where there were almost no cuts selected and we do not have a low amount of states at stages where too many cuts were selected. We used the cluster centers as the states to build the warm-start cuts, when solving the shifted problems.

Then we solved the model the shifted problem, that was the original problem with ξ following the shifted distribution: a uniform distribution in the integers between $1 - (t + 1)\%2$ and $5 - (t + 1)\%2$.

To compare the performance of the state pre-selection algorithm we observed the lower bound evolution with the iterations using both our method and the vanilla SDDP. As we can observer in figure 11 the lower bounds becomes get closer as we increase the number of iterations. But, with few iterations, the clustering warm-start method tends to perform better using the lower bound criteria. It is important to notice that the first iteration for our method represents an approximation with the warm-start cuts added, so we have more cuts in comparison with the vanilla SDDP.

We also used another comparison to observe the behavior of the approximations for the FCF's of each stage. In these tests we wanted to have global measure for how better is the whole approximation, that is not only observing the lower bound.

For these tests we used three approaches: The vanilla one, where there is no changes in SDDP; The SDDP algorithm with state pre-selection for warm-starting cuts; and finally one using random states as a warm-start. This last one, the random warm-start was an approach to verify if the selection done in the cluster warm-start is representing any difference.

In figure 12 we can observe the comparison between the evolution of the 1-norm of the first stage FCF as we add cuts through the algorithm. The blue curve represent the SPS clustering method, while the orange and green curves represent the vanilla SDDP and SPS random states selection respectively. We build these curves by computing the FCF values in a grid of state points and then we apply the 1-norm to the corresponding discretization. We calculated a grid for every 10 cuts added to the FCF's approximation, until reaching 400 cuts.

We observe that, for the first stage, using the SPS with clustering, yields a better result in comparison with the other 2 approaches. Comparing with normal SDDP the SPS with clustering is in average 0.03% higher and comparing with random SPS, the clustering approach, is 0.038% higher in average. As all those functions are lower approximations of the FCF of the true problem, so higher values means a closer approximation.

It is important to notice in figure 12 that the distance between the "Cluster" and "None" curves doesn't decrease significantly as new cuts are added.

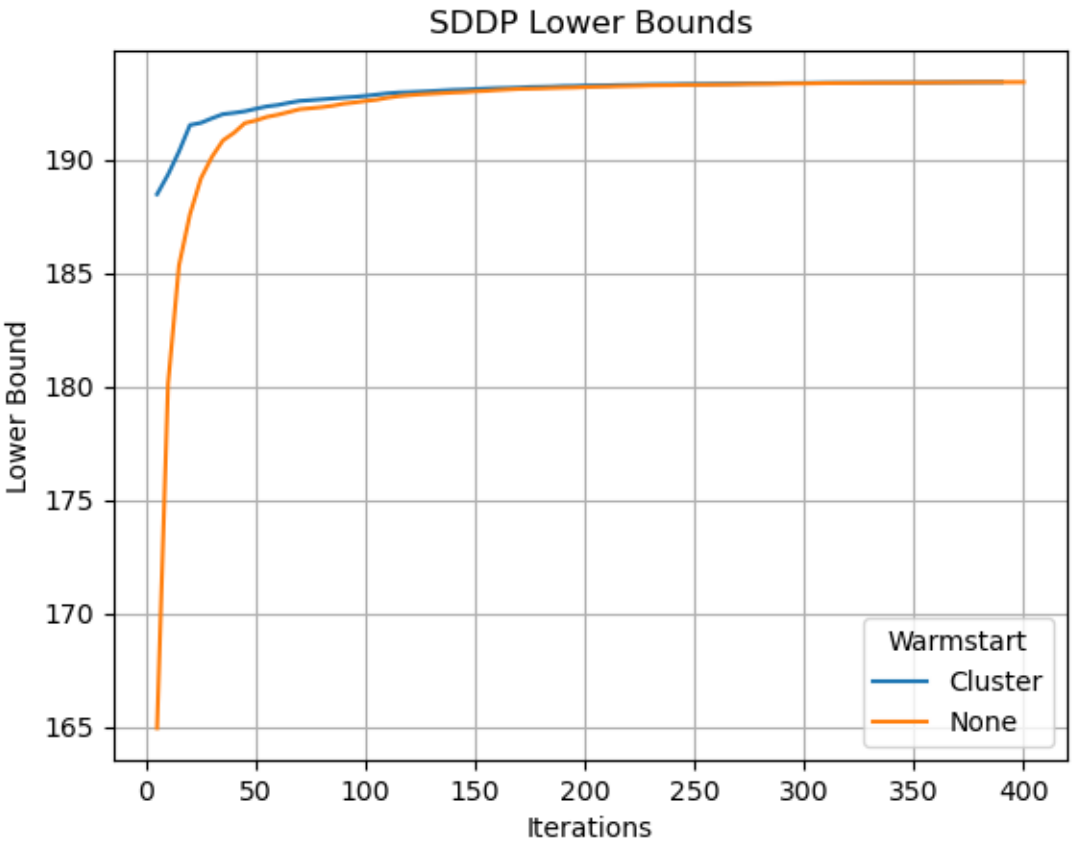


Figure 11: Lower bound using vanilla SDDP (orange curve) and cluster warm-start SDDP (blue curve).

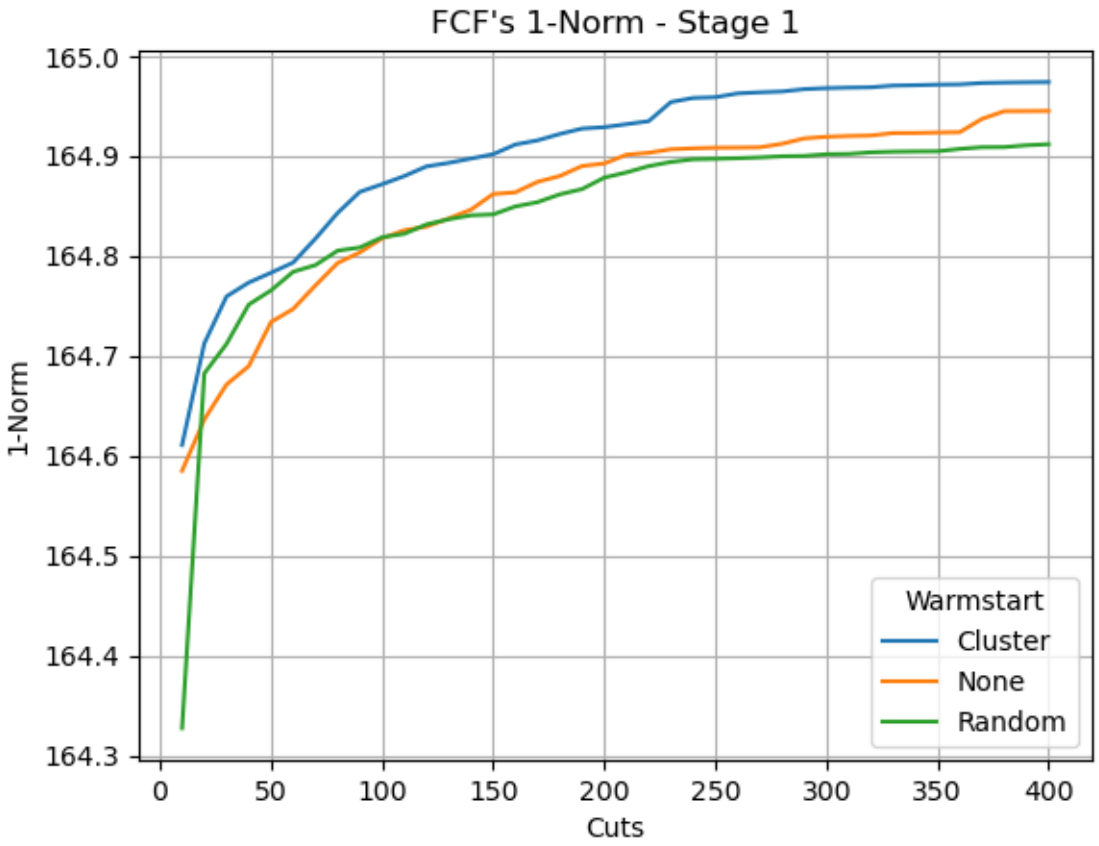


Figure 12: Random, cluster and no warm-start approaches comparison, with FCF's 1-norm by cut amount.

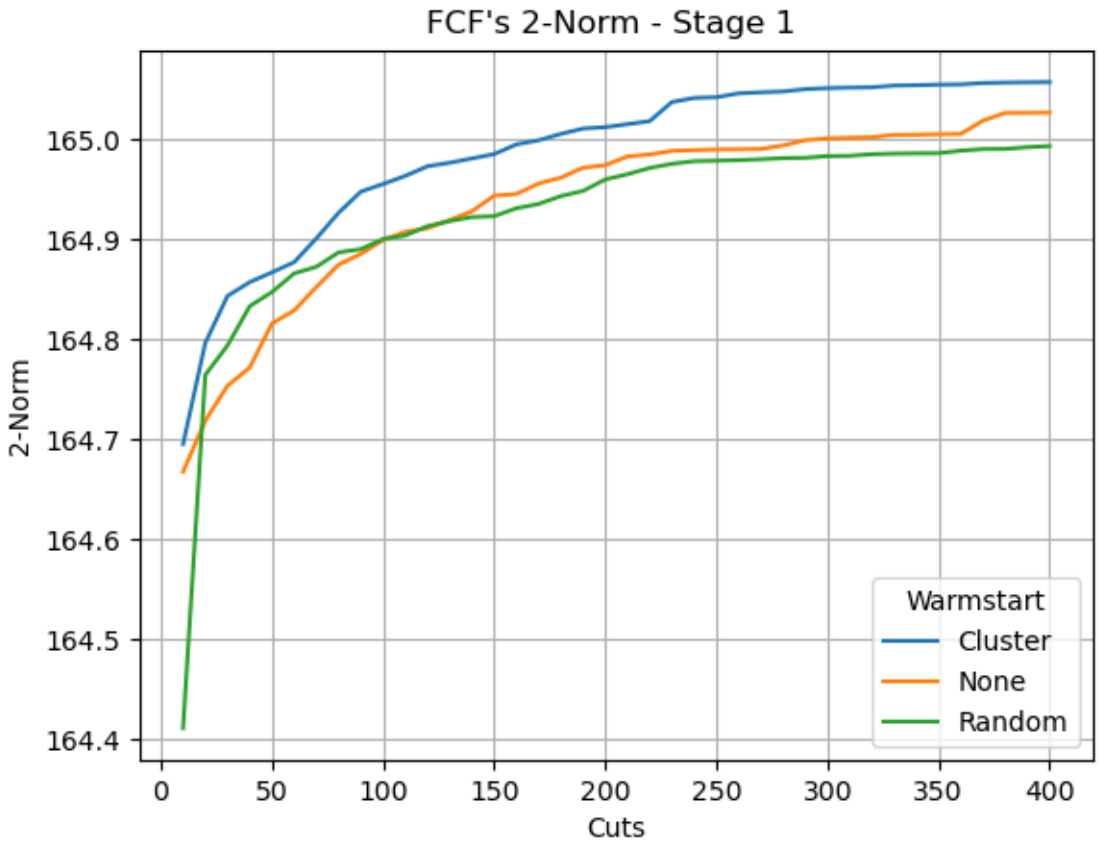


Figure 13: Random, cluster and no warm-start approaches comparison, with FCF's 2-norm by cut amount.

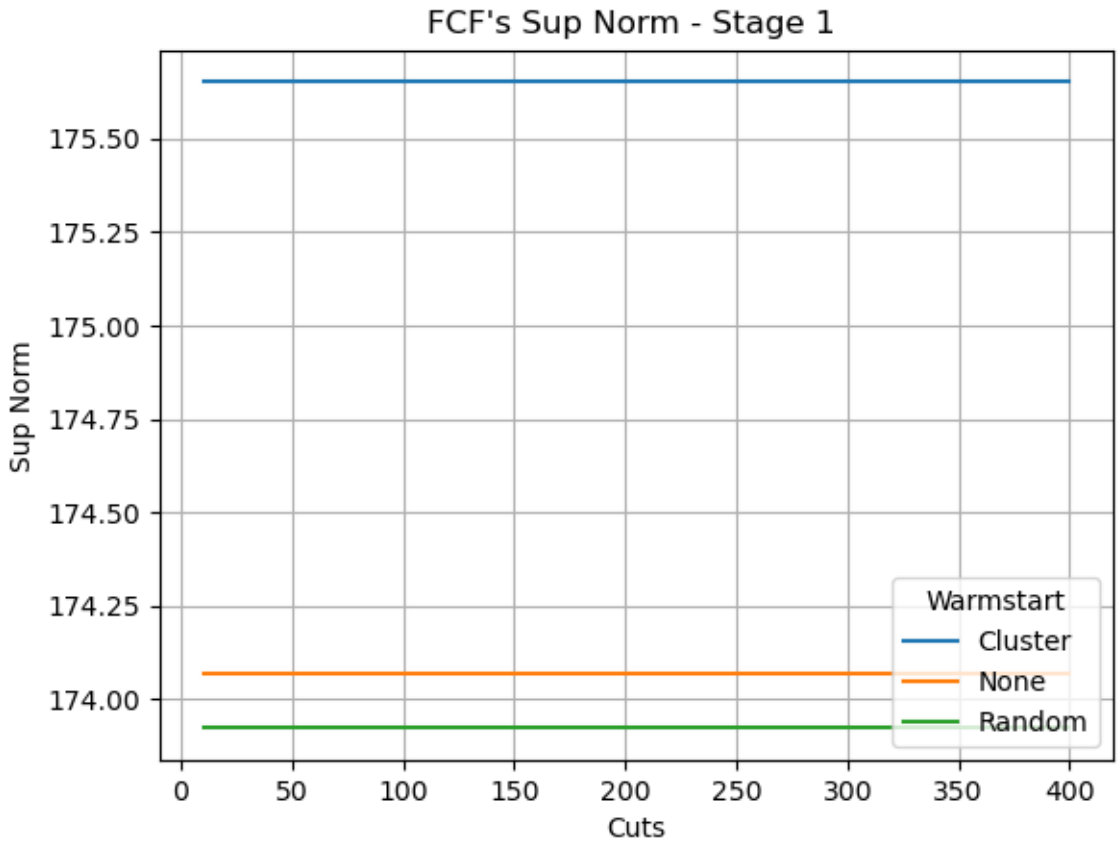


Figure 14: Random, cluster and no warm-start approaches comparison, with FCF's sup norm by cut amount.

One could expect that, as the algorithm progresses, both future cost function approximations would converge to the true one, and therefore the impact of the first nine cuts would decrease, which did not happen even after 400 cuts.

Also, in figures 13 and 14 we have this comparison calculated for the 2-norm and sup norm. And, independently of which metric was chosen, the cluster curve is still higher than the others at every point of the algorithm.

Furthermore, we calculated the same curves for the other stages FCF approximations. The intention is to verify if the SPS clustering approach presents a better result for the other FCFs too.

Observing figures 15, 16 and 17 we can see that for the second stage, it isn't clear if there is a better approach. Unlike the first stage, this graphs shows that for the 1-norm and 2-norm, depending on cuts number, the approach representing the better approximation changes. For the sup norm the cluster approach, as for the first stage, seems to present a better result, staying larger for almost every cut amount. This behavior, for the sup norm, may be caused by the way we select states. As we can remember, they are being selected at regions where the cuts present high dominance. But, to verify this hypothesis we would need more tests so we could assert this with certainty, like make SPS with states where the cuts of the original problem were built.

For the other stages, especially the last ones, the cluster approach does not outperform the other methods by any metric. But, in this work we are more interested in the early stage results, because the state pre-selection method usage is not optimized yet where the difference from shifted problem tends to be bigger.

In our current example, the first stage, or first month, of the shifted problem sees 19 months in the future, that are the other 19 stages. But this first month represents the second month of the original problem, where it sees only 18 months in the future. But for the 18-th month, the difference is seeing 3 months in the future instead of 2, and this implies on seeing 50% more future stages. So, it is reasonable that the FCF of the later months of the shifted problem are not so similar with the corresponding one of the original problem. At least we can claim that similarity isn't as for the earlier months. Then, we think that the states we selected are not necessarily the most significant for evaluating the corresponding FCF at latter stages.

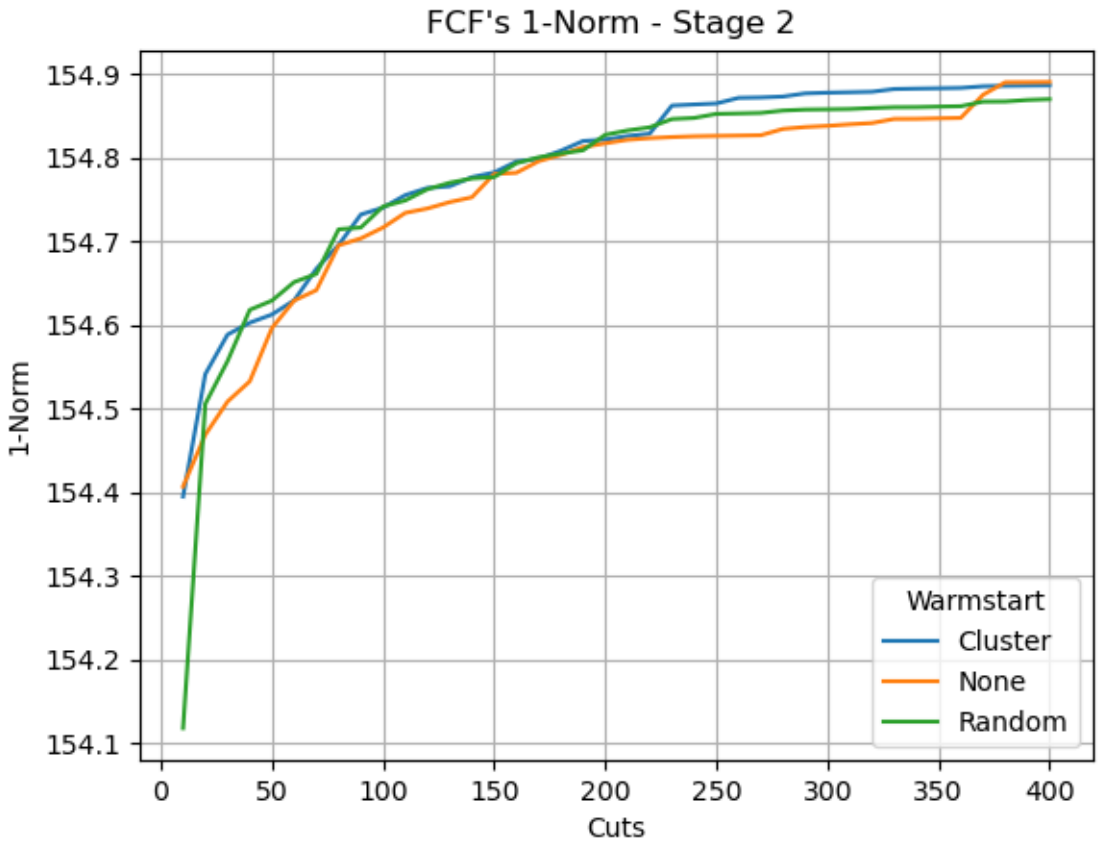


Figure 15: Random, cluster and no warm-start approaches comparison, with FCF's 1-norm by cut amount.

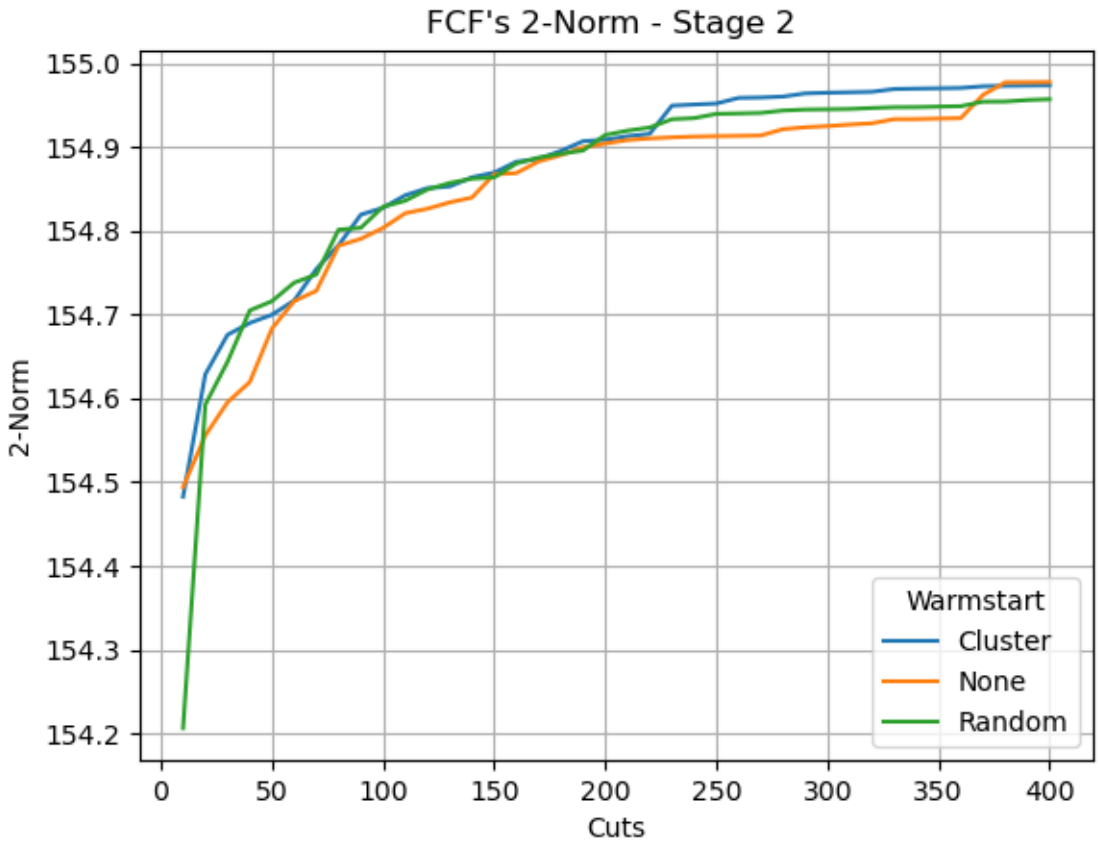


Figure 16: Random, cluster and no warm-start approaches comparison, with FCF's 2-norm by cut amount.

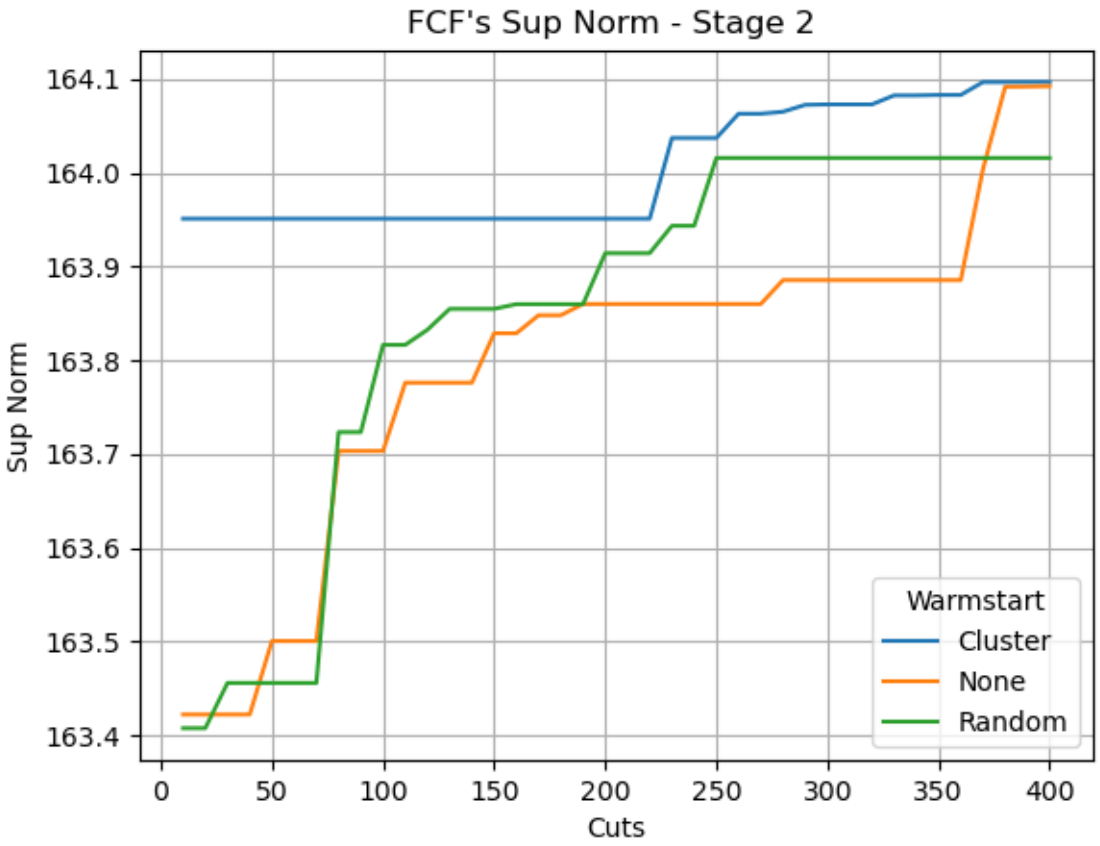


Figure 17: Random, cluster and no warm-start approaches comparison, with FCF's sup norm by cut amount.

7.2 Modified Models

We then modified some parameters of the problem, such as the demand and the power plants energy production capacity. This was done to verify the sensibility of the State Pre-Selection method. We can see in figures 31, 32, 33 and 34, in the appendix, that the cluster warm-start method outperform the no warm-start approach, specially with few cuts.

Following that, we also tried slightly changing the parameters *only for the shifted model*. This was a test on whether this method can be applied when the system is changing, as when there is an operation expansion planed for the further periods, like the addition of new power plants.

As we can observe, both in the lower bound test in the figures 18 and 19 and in the norm test in the figures 20 and 21, the overall results don't change too much. And, we can say, again, that with few cuts the cluster warm-start outperforms the no warm-start results.

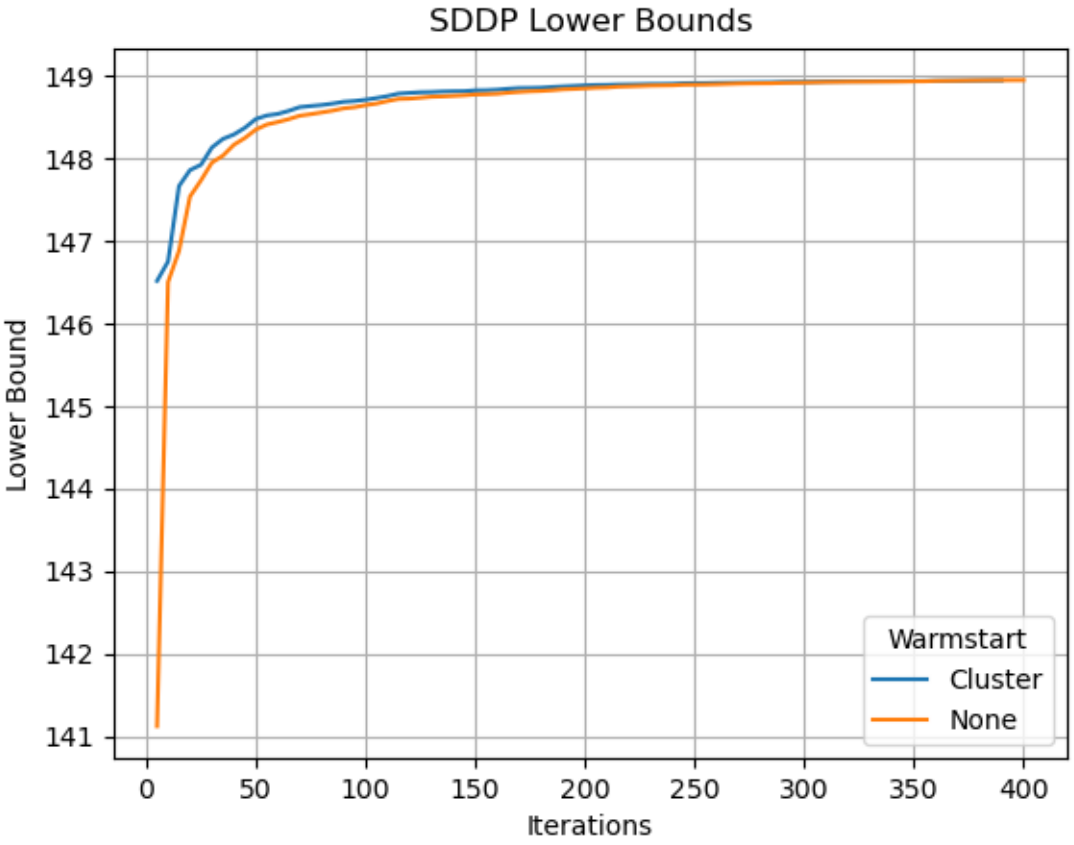


Figure 18: Lower bound comparison with vanilla SDDP and cluster warm-start SDDP. Changing the shifted problem parameters to: MaxTgen = 7; MaxHgen = (2,3); MaxVol = (7,9); demand = 11.

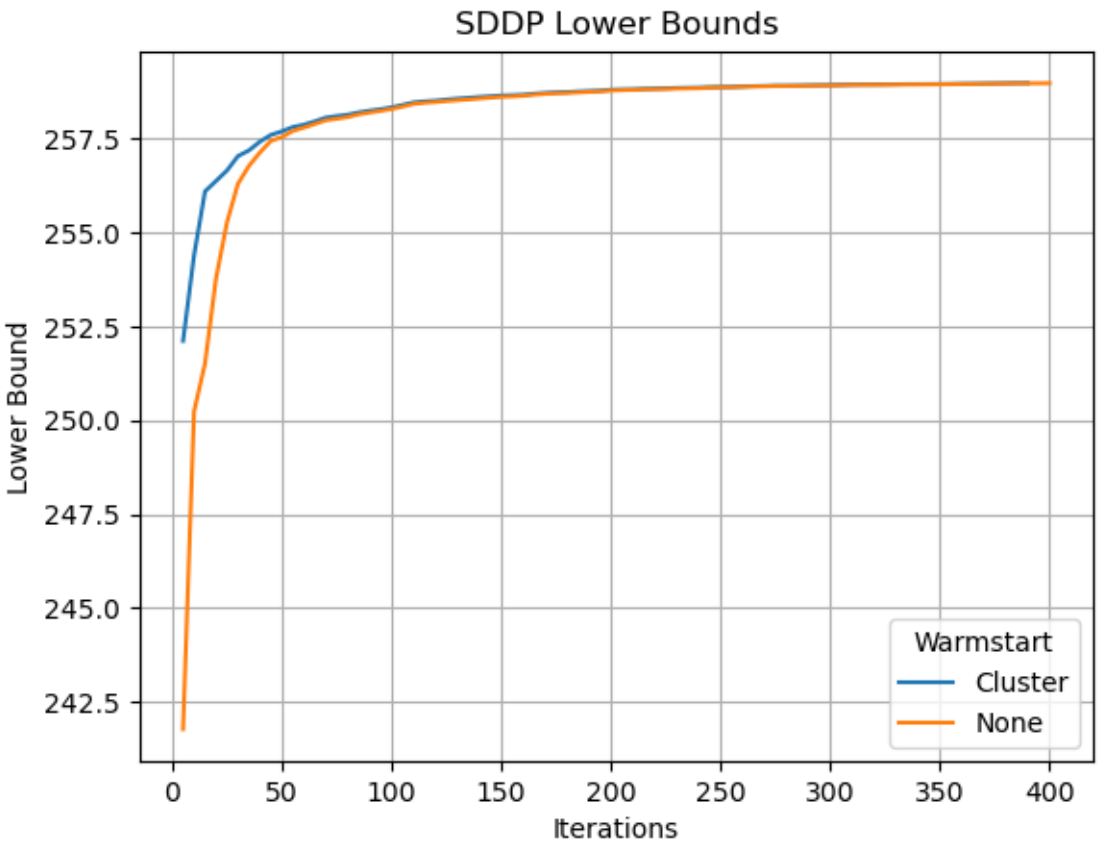
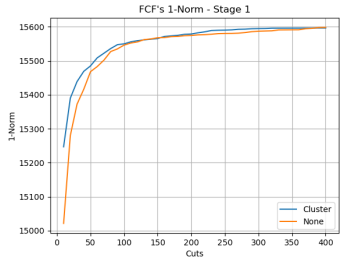
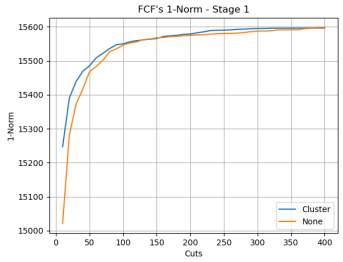


Figure 19: Lower bound comparison with vanilla SDDP and cluster warm-start SDDP. Changing the shifted problem parameters to: MaxTgen = 5; MaxHgen = (4,5); MaxVol = (7,7); demand = 9.

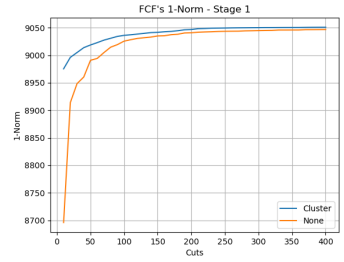


(a) 1-Norm

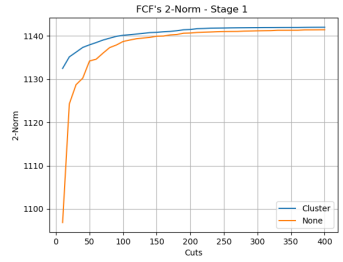


(b) 2-Norm

Figure 20: Perturbed problem: MaxTgen = 7; MaxHgen = (2,3); MaxVol = (7,9); demand = 11.



(a) 1-Norm



(b) 2-Norm

Figure 21: Perturbed problem: MaxTgen = 5; MaxHgen = (4,5); MaxVol = (7,7); demand = 9.

Finally, for this problem, we tried to use the states where the useful cuts were made, instead of using the state where it represented a better improvement. This change implied on having the clustering method presenting a performance difference at second and third stages similar to the one at the first stage. This can be observed in figures 22 and 23, where the blue curve remains above the other two, independently of the number of cuts performed. This could imply that having cuts where there were several good cuts being built in is more important than having cuts where there was need of improvement for the non shifted problem approximation.

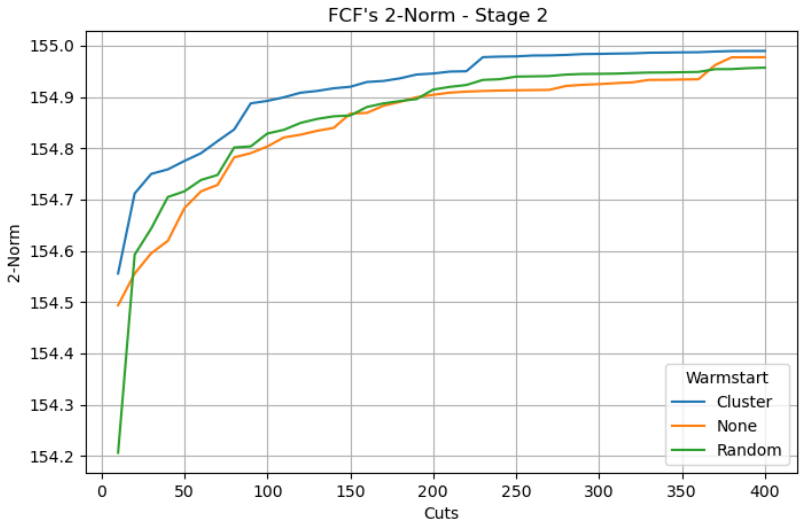


Figure 22: Random, cluster and no warm-start approaches comparison, with FCF's 2-norm by cut amount.

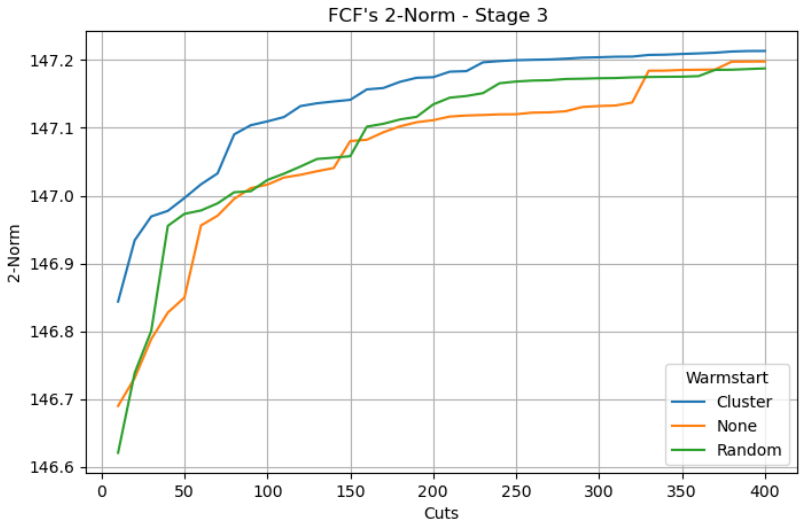


Figure 23: Random, cluster and no warm-start approaches comparison, with FCF's 2-norm by cut amount.

7.3 Model with 2 Aggregated Reservoirs

We then proceeded to test the State Pre-Selection algorithm with a larger problem instance. For the next results, we took a problem much closer to the Brazilian operation planning problem, but still taking in consideration only 2 aggregated reservoirs sub-systems (2-AR). In contrast with the previous model, this one uses real data for demand, the inflow distribution, the number and capacity of each thermal power plant as well as the reservoir volumes and hydro generation capacity.

For this model we had not only three thermal power plants, but hundreds. Also, the uncertainty is modeled based in the historical inflow series, instead of using just a discrete uniform with five possible outcomes. Other then that, we now consider transmission constraints, because the sub-systems can exchange energy.

Solving this model demanded significantly more computational effort, so we opted for only running a 5-stage time horizon. Also, as each stage sub-problems differs more significantly, we used a different clustering process for the states that will be used on the warm-start: Here we clustered each stage separately. That means that for the t-th stage of the shifted problem we used only the states clustered using the states selected from original problem $t + 1$-th stage.

By doing the clustering for each stage we select states that are more related with the each stage FCFs. This is important because this problem has less stages, so the shifted problem FCFs are not so similar to the original problem FCFs. This happens for the same reason we discussed earlier, for the latter stages of the example with 20 stages. Here, because we only have 5 stages, even for the first stage the shifted problem FCF is not so similar to the original problem second stage FCF.

Based on the previous results, we choose to pick the states where the useful cuts where built. We observed the lower bound curves in figure **??** and realized that the at the first iterations the cluster warm-start was not outperforming vanilla SDDP. In this graph we have that at the first iterations vanilla SDDP has a higher lower bound. But with a few more iterations the cluster warm-start start outperforming the vanilla algorithm. And, with enough iterations, both performances are the same.

We then observed the norm curves for all the 4 FCF's approximations, that are in appendix. By comparing these norms the warm-start, for first stage, do not outperform the Vanilla SDDP with few cuts. But, for stage 2, 3 and 4 the warm-start method clearly outperform the Vanilla SDDP with few cuts.

So, to have a local view of the approximations, to understand this behavior, we opted to observe locally the approximations, but only with 15 cuts to certify the SPS impact for the first iterations. The original problem still has 200 cuts.

In figure 25 we can observe the values difference between the clustering method and not using a warm-start. Where there is negative values we know that the warm-start method has a higher value. As we can observe, for stage one there is only a region where the difference is visible and there we have a pink region meaning a better warm-start solution. Also, the axis of this graphic

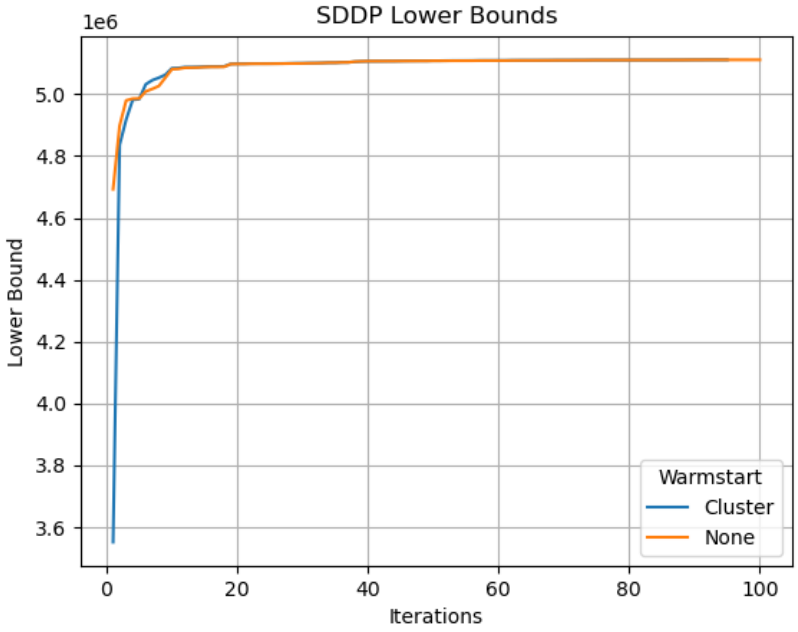


Figure 24: Lower bound curves for 2AR model. Blue curve: cluster warm-start; Orange curve: Vanilla SDDP.

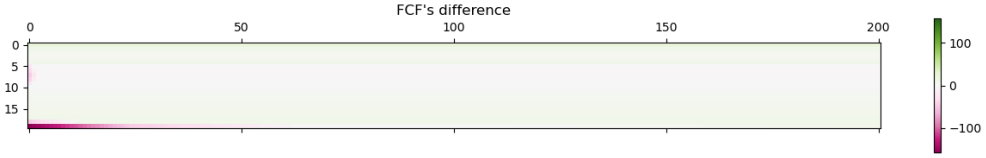


Figure 25: Difference between the cluster and none warm-start methods FCF's values, for first stage.

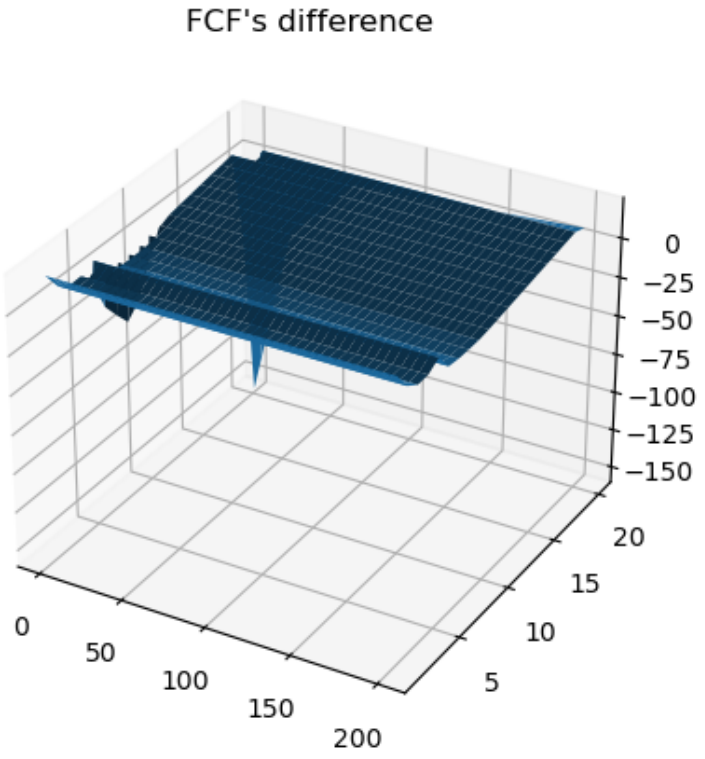


Figure 26: Difference between the curves of the FCF using the cluster and none warm-start methods, for first stage of the suifted model.

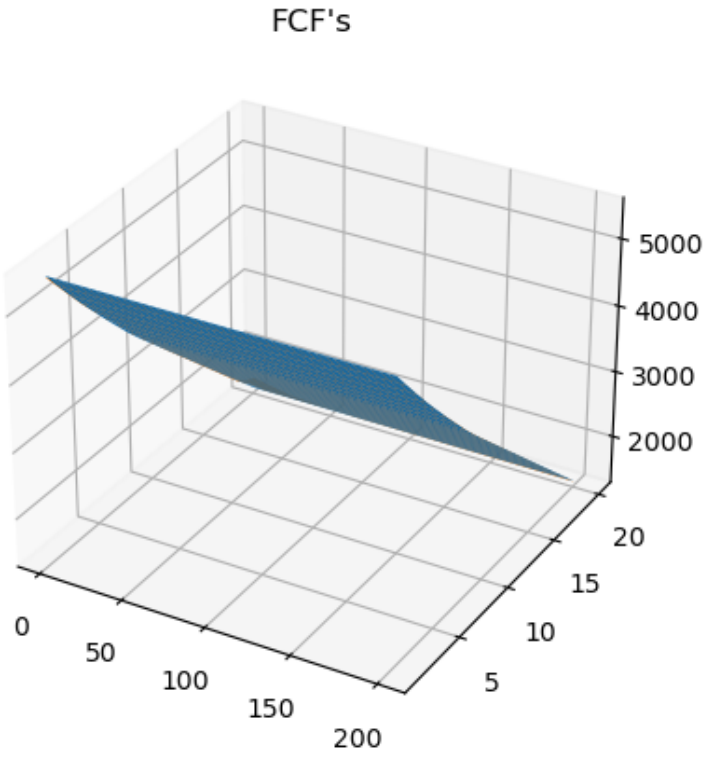


Figure 27: Curves of the FCF using Cluster and none warm-start methods, for first stage of the shifted model.

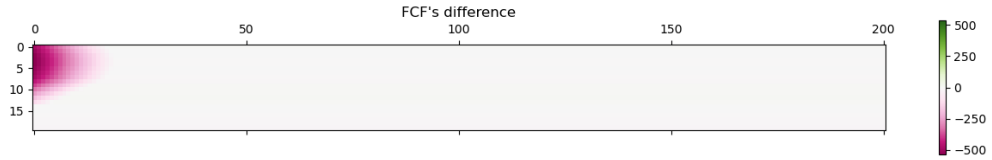


Figure 28: Difference between the values of the FCF with the cluster and none warm-start methods, for second stage of the shifted model.

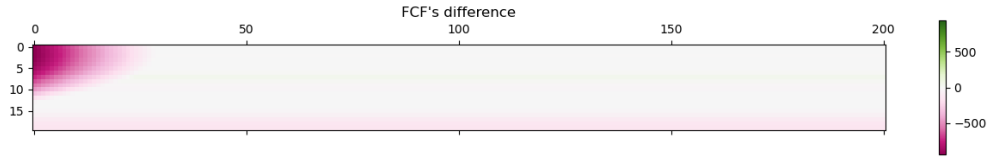


Figure 29: Difference between the values of the FCF with the cluster and none warm-start methods, for third stage of the shifted model.

and on the following ones mentioned in this section, are divided by 1000 only to help exposing the results.

To visualise more clearly I have also plotted the surface 26 that is the $3D$ version 25. Here we can see that there is some regions where the clustering method is worse, but the only region where the difference is relevant is indeed in warm-start favor. Also, I've plotted both methods FCFs at 26, to see if both have the same shape and the figure shows that they have.

So, for the first stage, we can say that there is an improvement by using warm-start even for the 2-AR model. Because, despite the fact that warm-start do not improve everywhere, we see in 26 and 25 that where there is improvement it is a higher improvement.

Also, in the previous model we observed that there was a tendency of lower impact for the clustering warm-start method for the further stages. But, as the reader can observe from figures 28, 29 and 30 that, as in the FCF norm graphics, the State Pre-Selection impact was more evident at the further stages then it was at the first one. This is much clear at 29 where even at the bottom white part there is a tendency to the pink. Also, it is important to observe in 37 (in the appendix) that, at the third stage, the FCF's values are between 0 and $2.8e+3$. So, an $8e+2$ difference in 40 (also in the appendix), is very significant, specially at the region near volumes at zero. That region is a critical place to have good approximation, because that is where we have the higher values of the FCF. So, having a good approximation there can help avoid a more expensive operation.

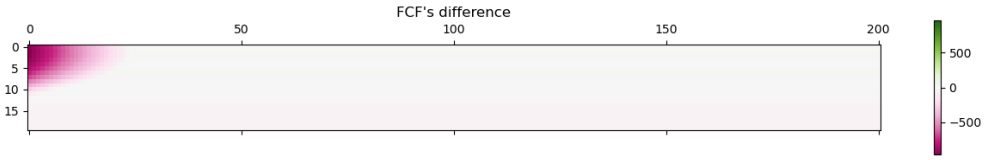


Figure 30: Difference between the values of the FCF with the cluster and none warm-start methods, for fourth stage of the shifted model.

8 Conclusion and Future Works

After testing the SPS method and comparing it with the Vanilla SDDP and with random states warm-start, we can conclude a few things. First, the usage of random states does not match the method proposed. In fact, when there are low amount of cuts it is the worst method being significantly behind even of Vanilla SDDP. This was expected, but was good to show that trying to find the right states is one of the important ideas.

Also, by observing this results we can say that looking for regions where there wasn't enough improvement may be a blind strategy. By ignoring the states where the useful cuts were built in and only looking to the states where they presented the highest dominance we could be missing the aid that this cuts presents over the other places at the feasible set. One idea that can be explored in future works is to mix these two strategies, with that we would collect twice more states and they would be more spread.

Other then that, at the second model we observed the impact of clustering individually for each stage can be significant to have better approximations at later stages. It would take more tests to certify that this difference observed on the more realistic result comes with that method change. Also, we could try using other methods to select the state regions. Instead of using K-means we could pair the parameters with states visited in previous solved problems and then, when solving the shifted problem, use the new parameters to identify which states must be selected.

A last proposal is to use the State Pre-Selection method in other kind of models and verify if the impacts are different. In particular, the periodic infinite horizon is an interesting model to be tested. For this kind of models we have that the future stages share the same FCF with the earlier stages. So, by having the capacity of improve the earlier stages FCF, as observed, we may improve the later FCF to, because they are the same.

9 Appendix

9.1 Graphics for Problems With Different Parameters

This section includes the numerical tests done with the simple models exposed at section 7.1. But we changed the demand and the hydro and thermal maximum generation slightly to verify if the warm-start method outperforms the Vanilla SDDP for a slightly different problem too.

Remember that we use to have the following values for the model parameters:

- $c = (2, 5, 10)$;
- $MaxTgen = (6, 6, 6)$;
- $MaxHgen = (3, 4)$;
- $MaxVol = (6, 8)$;
- $vol_0 = (3, 4)$;
- $demand = 10$.

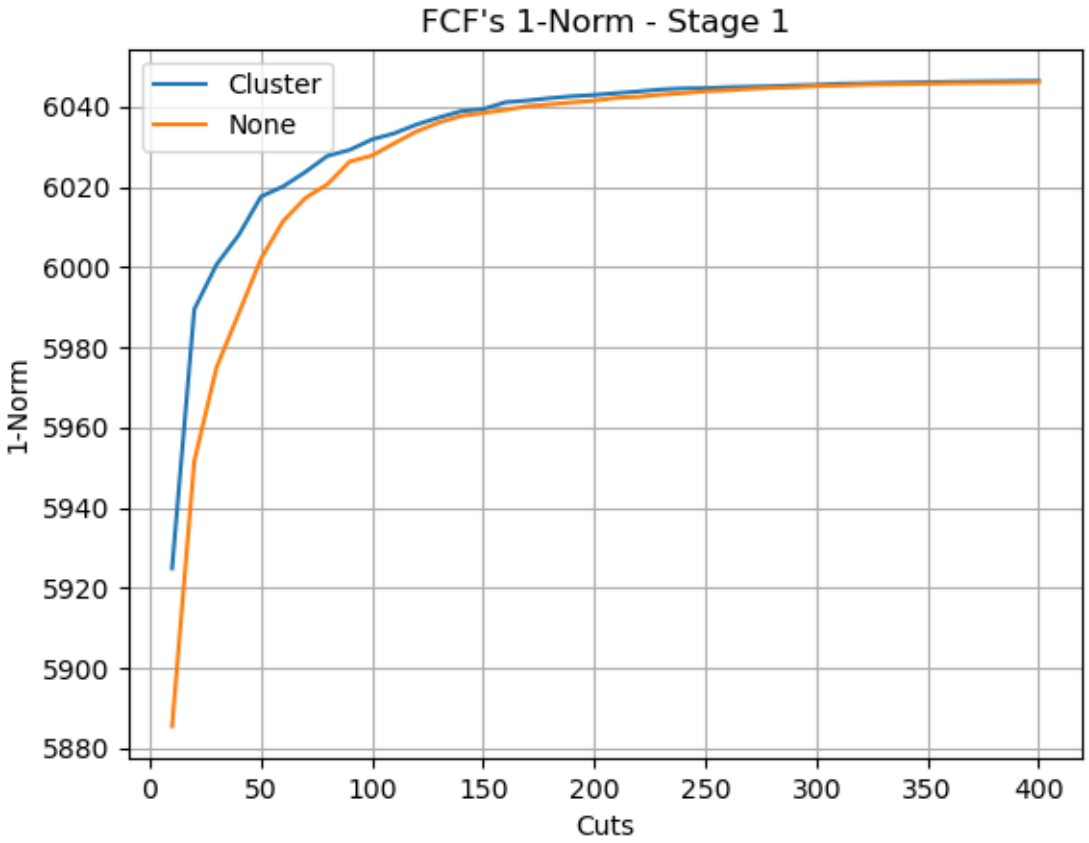


Figure 31: Model with demand = 9.

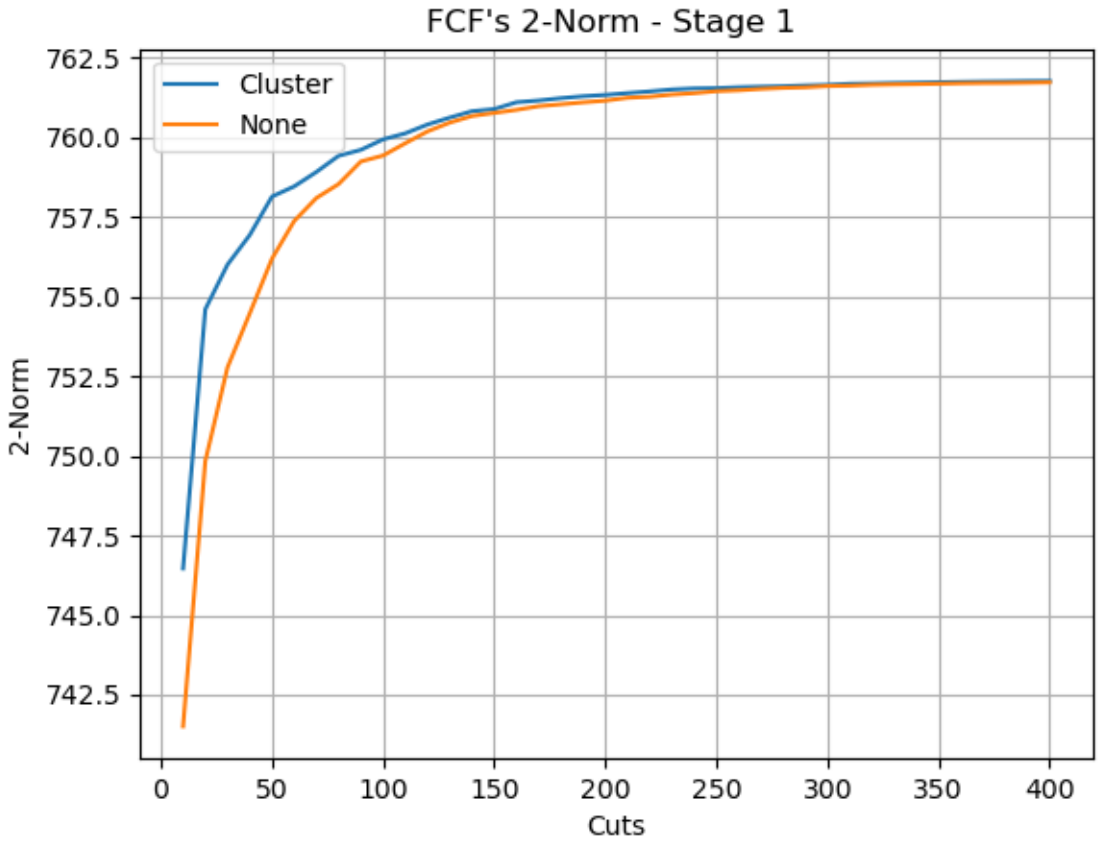


Figure 32: Model with demand = 9.

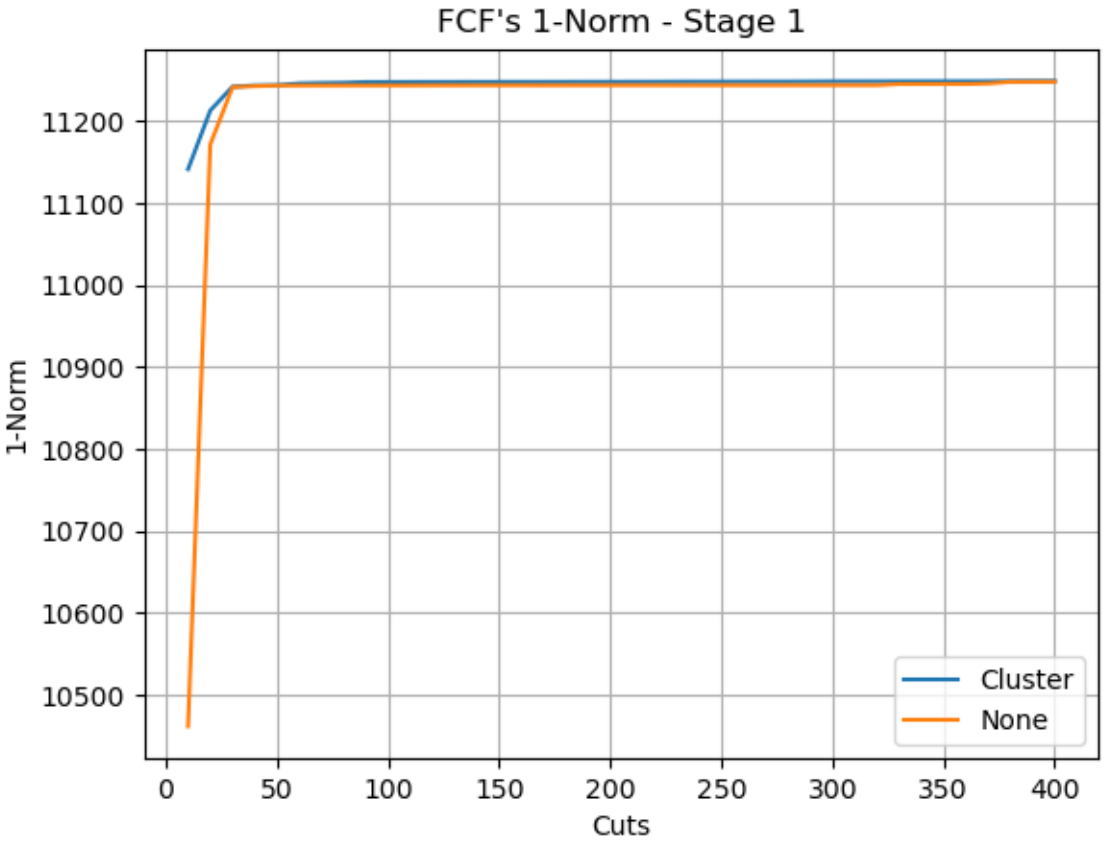


Figure 33: Model with MaxTgen = 7 and MaxHgen = (4,5).

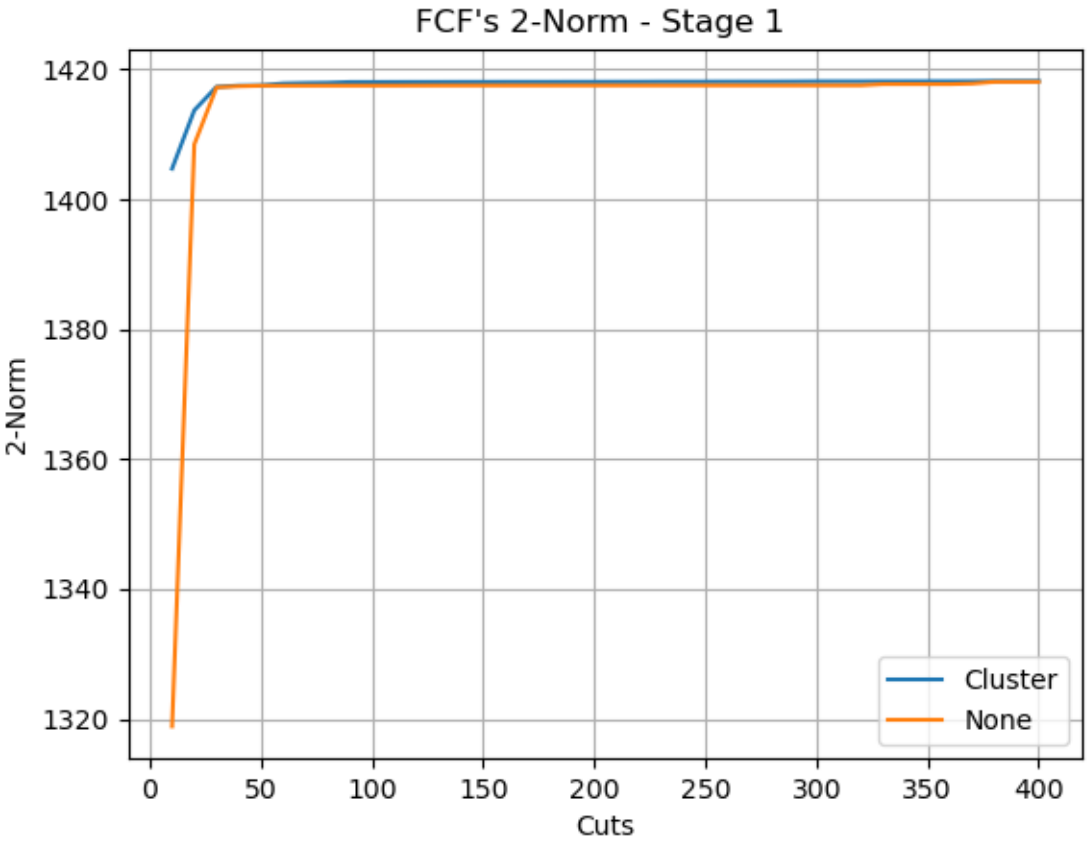
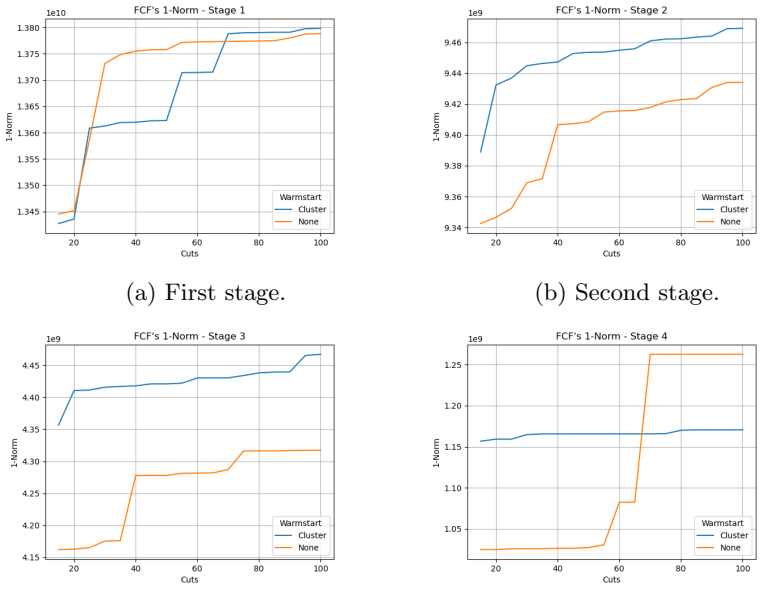


Figure 34: Model with MaxTgen = 7 and MaxHgen = (4,5).

9.2 Graphs of the 1-norm in the 2 reservoir model

For completion, we include the evolution of the 1-norm for the FCF's in problems of section 7.3, comparing the use of warm-start methods.



(a) First stage.

(b) Second stage.

(c) Third stage.

(d) Fourth stage.

Figure 35: 1-norm for the Future Cost Functions

9.3 Curves for 2-AR Model

Here we have the curves of the FCFs stages 2, 3 and 4, that was presented for the first stage previously.

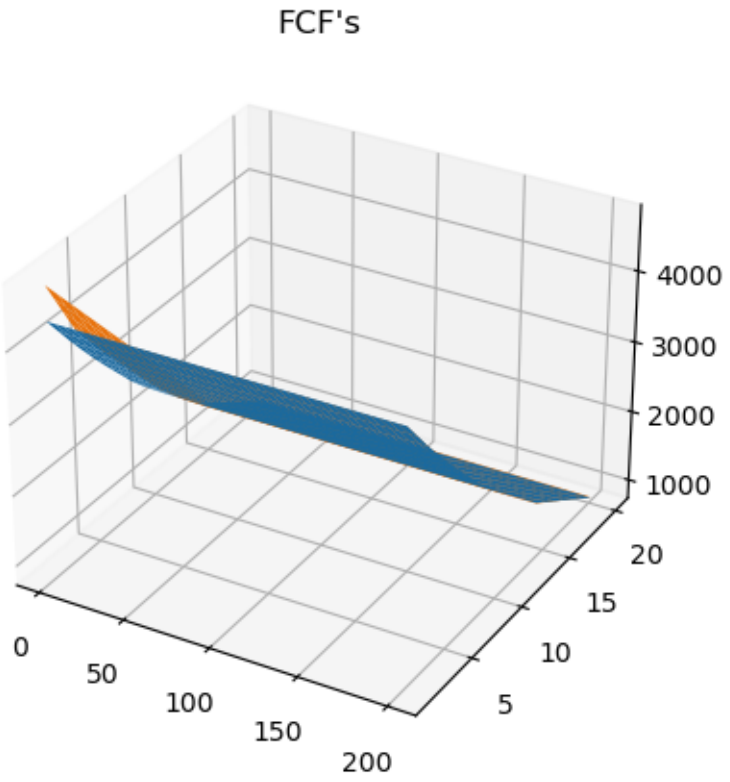


Figure 36: Curves of the FCF using Cluster and none warm-start methods, for second stage of the shifted model.

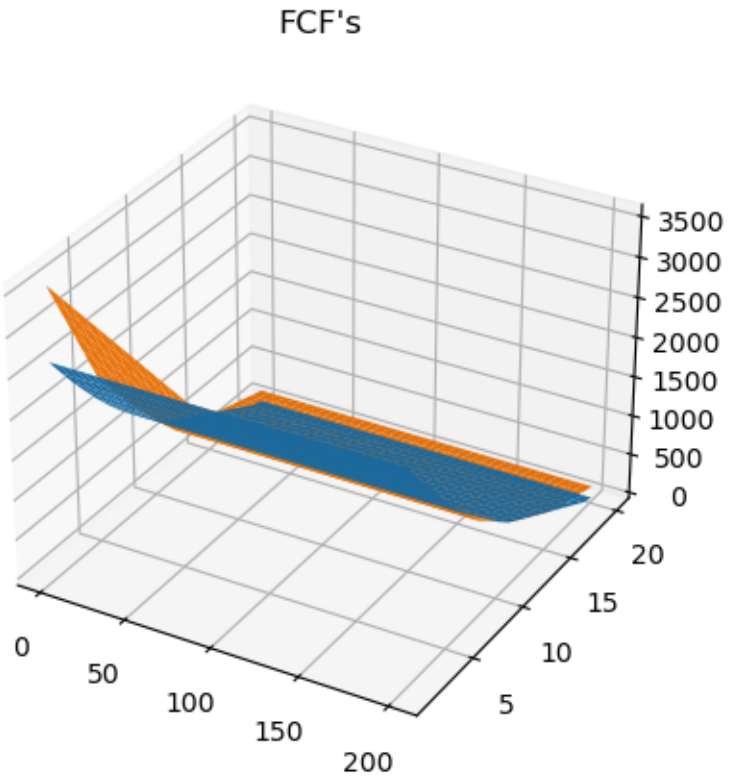


Figure 37: Curves of the FCF using Cluster and none warm-start methods, for third stage of the shifted model.

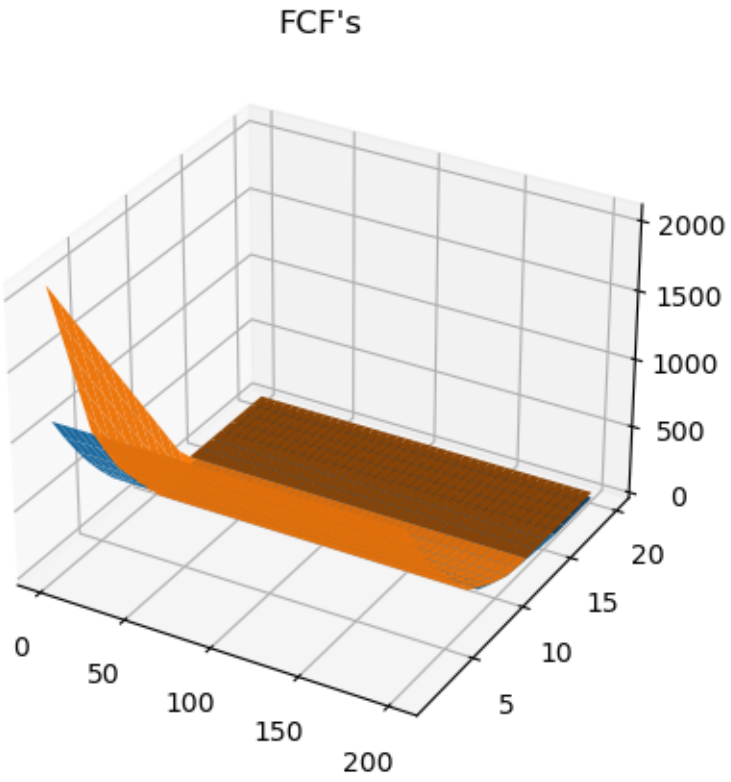


Figure 38: Curves of the FCF using Cluster and none warm-start methods, for fourth stage of the shifted model.

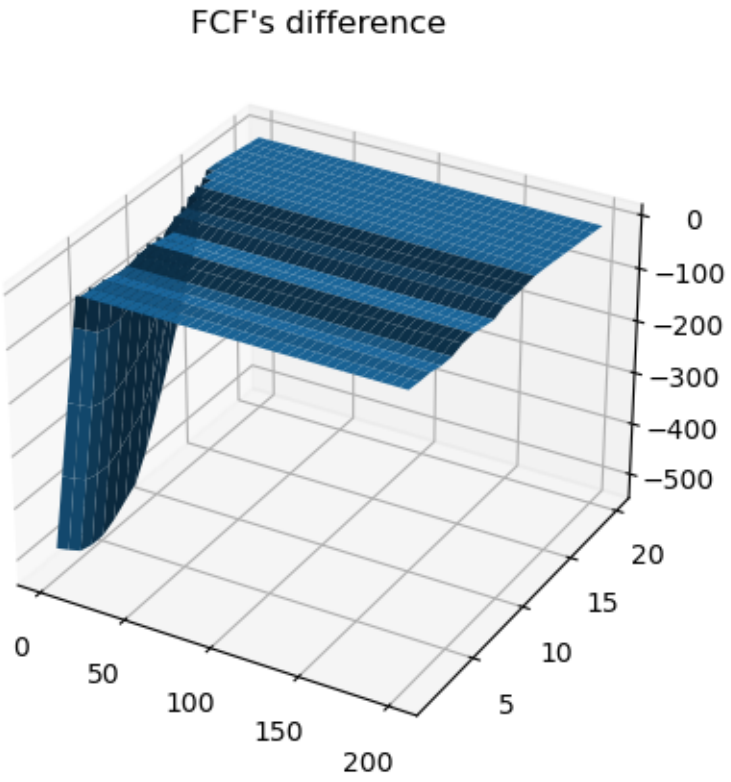


Figure 39: Difference between the curves of the FCF using the cluster and none warm-start methods, for second stage of the shifted model.

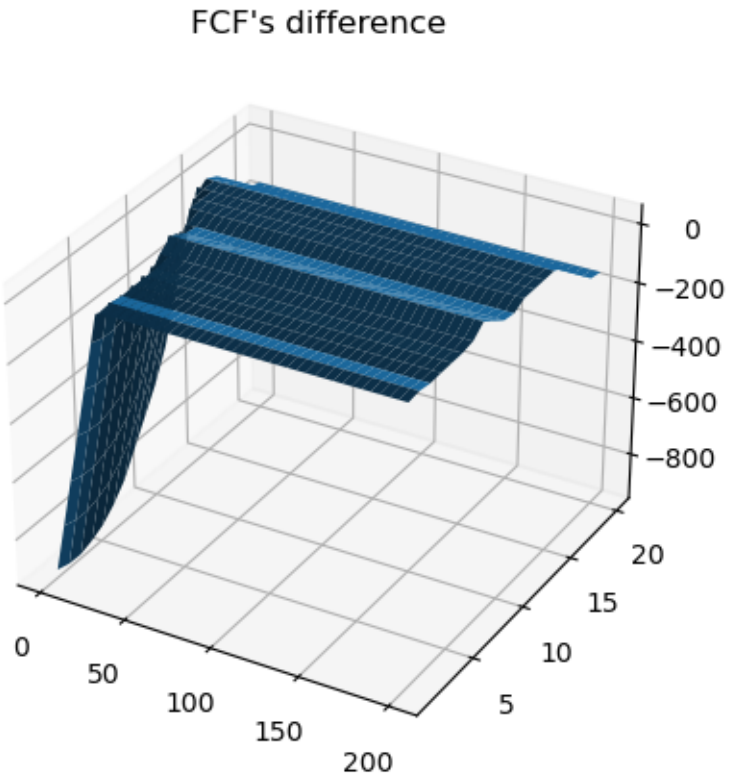


Figure 40: Difference between the curves of the FCF using the cluster and none warm-start methods, for third stage of the shifted model.

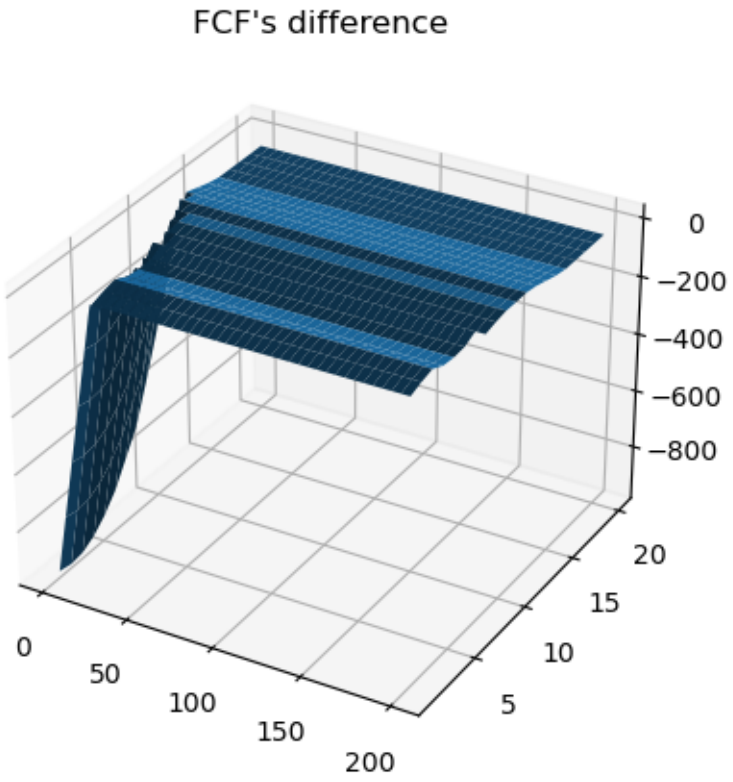


Figure 41: Difference between the curves of the FCF using the cluster and none warm-start methods, for fourth stage of the shifted model.

References

[ACF20] Shabbir Ahmed, Filipe Goulart Cabral, and Bernardo Freitas Paulo da Costa. "Stochastic Lipschitz dynamic programming". In: *Mathematical Programming* (2020), pp. 1–39.

[Bir85] John R Birge. "Decomposition and partitioning methods for multistage stochastic linear programs". In: *Operations research* 33.5 (1985), pp. 989–1007.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CS87] MF Carvalho and Secundino Soares. "An efficient hydrothermal scheduling algorithm". In: *IEEE Transactions on Power Systems* 2.3 (1987), pp. 537–542.

[DHL17] Iain Dunning, Joey Huchette, and Miles Lubin. "JuMP: A Modeling Language for Mathematical Optimization". In: *SIAM Review* 59.2 (2017), pp. 295–320. DOI: 10.1137/15M1020575.

[DK17] Oscar Dowson and Lea Kapelevich. "SDDP.jl: a Julia package for stochastic dual dynamic programming". In: *Optimization Online* (2017). URL: http://www.optimization-online.org/DB_HTML/2017/12/6388.html.

[Dow20] Oscar Dowson. "The policy graph decomposition of multistage stochastic programming problems". In: *Networks* 76.1 (2020), pp. 3–23.

[DPF15] Vitor L De Matos, Andy B Philpott, and Erlon C Finardi. "Improving the performance of stochastic dual dynamic programming". In: *Journal of Computational and Applied Mathematics* 290 (2015), pp. 196–208.

[GLP13] P Girardeau, V Leclère, and Andy B Philpott. "On the Convergence of Decomposition Methods for Multistage Stochastic Convex Programs". In: (2013).

[Gor+92] BG Gorenstin et al. "Stochastic optimization of a hydro-thermal system including network constraints". In: *IEEE Transactions on Power Systems* 7.2 (1992), pp. 791–797.

[Gui+17] Vincent Guigues et al. "Dual Dynamic Programing with cut selection: Convergence proof and numerical experiments". In: *European Journal of Operational Research* 258.1 (2017), pp. 47–57.

[HE03] Greg Hamerly and Charles Elkan. "Learning the k in k-means". In: *Advances in neural information processing systems* 16 (2003).

[Hun07] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[PD12] Andrew B Philpott and Vitor L De Matos. “Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion”. In: *European Journal of operational research* 218.2 (2012), pp. 470–483.

[PG08] Andrew B Philpott and Ziming Guan. “On the convergence of stochastic dual dynamic programming and related methods”. In: *Operations Research Letters* 36.4 (2008), pp. 450–455.

[PMF13] Andy Philpott, Vitor de Matos, and Erlon Finardi. “On solving multistage stochastic programs with coherent risk measures”. In: *Operations Research* 61.4 (2013), pp. 957–970.

[PP85] MVF Pereira and LMVG Pinto. “Stochastic optimization of a multireservoir hydroelectric system: A decomposition approach”. In: *Water resources research* 21.6 (1985), pp. 779–792.

[PP91] Mario VF Pereira and Leontina MVG Pinto. “Multi-stage stochastic optimization applied to energy planning”. In: *Mathematical programming* 52.1 (1991), pp. 359–375.

[RG92] TA Rotting and A Gjelsvik. “Stochastic dual dynamic programming for seasonal scheduling in the Norwegian power system”. In: *IEEE Transactions on Power Systems* 7.1 (1992), pp. 273–279.

[Ros81] Richard E Rosenthal. “A nonlinear network flow algorithm for maximization of benefits in a hydroelectric power system”. In: *Operations research* 29.4 (1981), pp. 763–786.

[SD20] Alexander Shapiro and Lingquan Ding. “Periodical multistage stochastic programs”. In: *SIAM Journal on Optimization* 30.3 (2020), pp. 2083–2102.

[Sha+13] Alexander Shapiro et al. “Risk neutral and risk averse stochastic dual dynamic programming method”. In: *European journal of operational research* 224.2 (2013), pp. 375–391.

[Sis] Operador Nacional do Sistema Elétrico. *ONS - Histórico-da-Operação*. `http://www.ons.org.br/Paginas/resultados-da-operacao/historico-da-operacao/geracao_energia.aspx`, accessed 2022-11-02.

[SYR13] Archana Singh, Avantika Yadav, and Ajay Rana. “K-means with three different distance metrics”. In: *International Journal of Computer Applications* 67.10 (2013).

[ZAS19] Jikai Zou, Shabbir Ahmed, and Xu Andy Sun. “Stochastic dual dynamic integer programming”. In: *Mathematical Programming* 175.1 (2019), pp. 461–502.

[ZS22] Shixuan Zhang and Xu Andy Sun. “Stochastic dual dynamic programming for multistage stochastic mixed-integer nonlinear optimization”. In: *Mathematical Programming* (2022), pp. 1–51.